VHDL satelliti telecomunicazioni robotica Mr A. Keer radiofrequenza timers multitasking REAL TIME motori DC controllo radio alimentatori BJT interruttore capacitivo didattica controllo relè USB multiplexer tutorial

n. 288 GIUGNO - Anno XXV - € 6,00

Fe fare elettronica

www.farelettronica.com

CHIAVE a combinazione elettronica

ESCLUSIVO! costruzione di una TESTA ROBOTICA

PROGETTI

PULSANTE CON SENSORE CAPACITIVO

CONTROLLO RELÈ VIA USB

GENERATORE RF DA 30MHz

Sistemi REAL TIME cosa sono e come funzionano

ISSN 1591-2272
90288
9 771591 227008

MENSILE Poste Italiane Spa - Spedizione in abbonamento Postale - D.L. 353/2003 (conv. In L. 27/02/2004 n. 46) art. 1, comma1, DCB Milano. In caso di mancato recapito, restituire all'editore che si impegna a pagare la relativa tassa presso il CPM di Roserio - Milano

Sommario

288 giugno 2009

DIVERTITI E METTI ALLA PROVA LE TUE CONOSCENZE CON **ELETTRO QUIZ** E VINCI OGNI MESE ESCLUSIVI PREMI!

GLI ARTICOLI CONTRASSEGNATI COL SIMBOLO → SONO GIÀ DISPONIBILI IN FORMATO PDF* ALL'INDIRIZZO
www.farelettronica.com/club
*Puoi iscriverti al CLUB di Fare Elettronica versando una piccola quota annuale.

elenco inserzionisti

di MAURIZIO DEL CORSO

L'invenzione della chiave risale a oltre 3000 anni fa, epoca in cui con un bastone di legno dotato di cavicchi era possibile spostare un chiavistello su una grossa porta di legno. La grande evoluzione di questa prestigiosa invenzione ci ha portato ai più recenti modelli a combinazione elettronica o ad impronta digitale ed oggi con un minimo di conoscenze in elettronica è addirittura possibile auto-costruirne una

Per non perdere LE CHIAVI

La chiave. A chi non è successo di perderla? L'unico rimedio per i più distratti è quello di usarne una elettronica. Nonostante vi siano in commercio diversi modelli, abbiamo pensato di proporvi il progetto di una chiave a combinazione elettronica realizzata con un microcontrollore e pochi altri componenti aggiuntivi. Per noi e i nostri lettori il gusto del "fai da te" ha sempre il sopravvento per cui realizzare la propria serratura e magari personalizzare il firmware per aumentare la sicurezza, non ha prezzo. Mettete quindi al sicuro il vostro laboratorio senza il rischio di perderne le chiavi: vi basterà ricordare solo la combinazione e digitarla sul keypad. Tuttavia l'uso del tastierino numerico non è il massimo della sicurezza: digitando sempre la stessa combinazione i tasti interessati si logoreranno più degli altri, per cui il compito dell'eventuale hacker sarà facilitato in quanto le cifre coinvolte nella combinazione saranno da ricercarsi solo tra i tasti più logori. Per ovviare a questo inconveniente è dunque opportuno usare serrature a comando vocale o a riconoscimento di impronta digitale. Procediamo per gradi, tratteremo successivamente queste tecnologie. Intanto non dimenticate la combinazione e non dimenticate di dotare il circuito di una batteria ricaricabile in tampone per non restare chiusi fuori in caso di blackout!

DIRETTORE RESPONSABILE
Antonio Cirella

DIRETTORE TECNICO
Maurizio Del Corso

Comitato Scientifico
Simone Masoni (Microtest), Francesco Picchi (Microtest), Massimo Rovini (Università degli Studi di Pisa), Tiziano Galizia (Tigal), Claudio Turchetti (Università Politecnica delle Marche).

Art Director
Patrizia Villa

Hanno collaborato in questo numero:
Nicola De Crescenzo, Francesco Di Lorenzo, Antonio Giannico, Massimiliano Miocchi, Giorgio Ober, Francesco Pentella, Federico Pinto, Roberto Prestianni, Remo Riglioni, Vincenzo Sorce, Franco Tedeschi.

Direzione Redazione
Pubblicitá
International Advertisement
INWARE Edizioni srl
Via Cadorna, 27/31
20032 Cormano (MI)
Tel. 02.66504755
Fax 02.66508225
info@inwaredizioni.it
www.inwaredizioni.it
Redazione: fe@inwaredizioni.it

Stampa
Grafiche Sima
Via Provinciale Francesca, 52
24040 Ciserano (BG)

Distribuzione
Parrini & C s.p.a.
Via di Santa Cornelia, 9
00060 Formello (RM)

Ufficio Abbonamenti
INWARE Edizioni srl
Via Cadorna, 27/31
20032 Cormano (MI)
Per informazioni, sottoscrizione o rinnovo dell’abbonamento:
abbonamenti@inwaredizioni.it
Tel. 02.66504755
Fax. 02.66508225
L'ufficio abbonamenti è disponibile telefonicamente dal lunedì al venerdì dalle 14,30 alle 17,30.
Tel. 02.66504755
Fax 02.66508225

Abbonamento per l’Italia:
€ 49,50
Abbonamento per l’estero:
€ 115,00
Gli arretrati potranno essere richiesti, per iscritto, a **€ 9,00** oltre le spese di spedizione

Autorizzazione alla pubblicazione
Tribunale di Milano n.647
del 17/11/2003

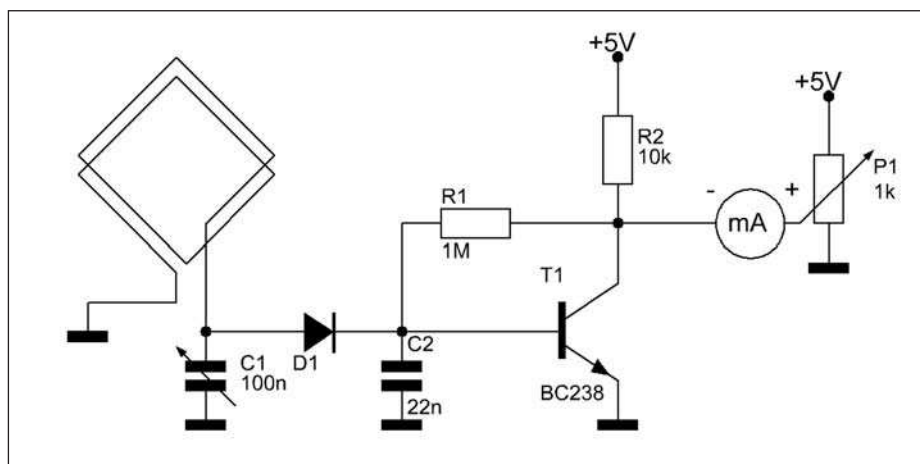
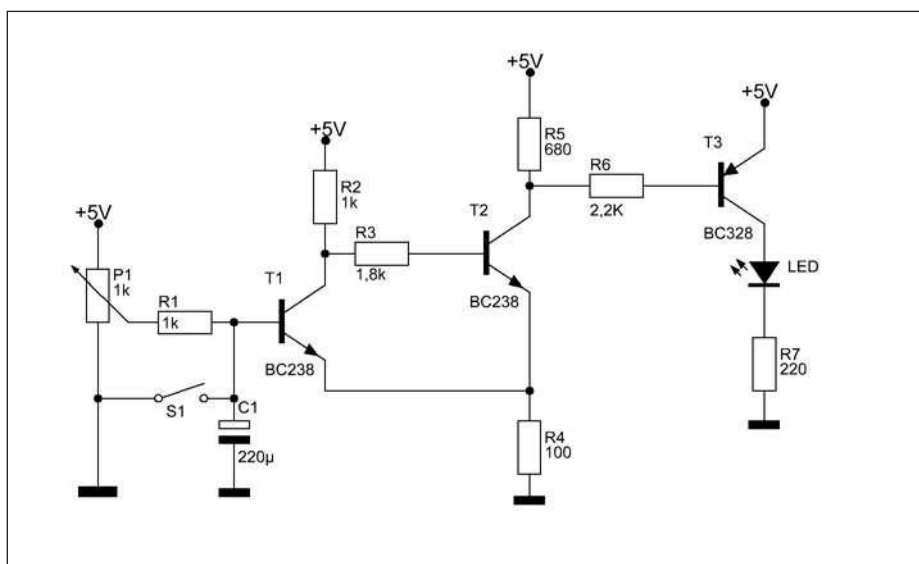
Collaborare con FARE ELETTRONICA
Le richieste di collaborazione vanno indirizzate all’attenzione di Maurizio Del Corso (m.delcorso@inwaredizioni.it) e accompagnate, se possibile, da una breve descrizione delle vostre competenze tecniche e/o editoriali, oltre che da un elenco degli argomenti e/o progetti che desiderate proporre.

di FRANCESCO DI LORENZO

LAMPEGGIATORE CONTROLLATO CON LA LUCE

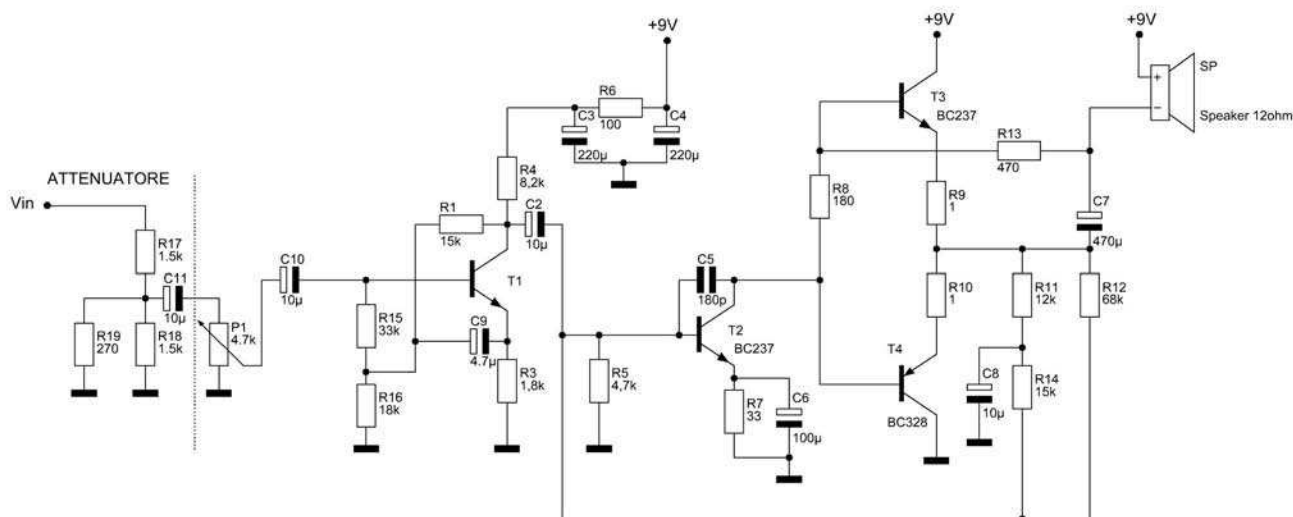
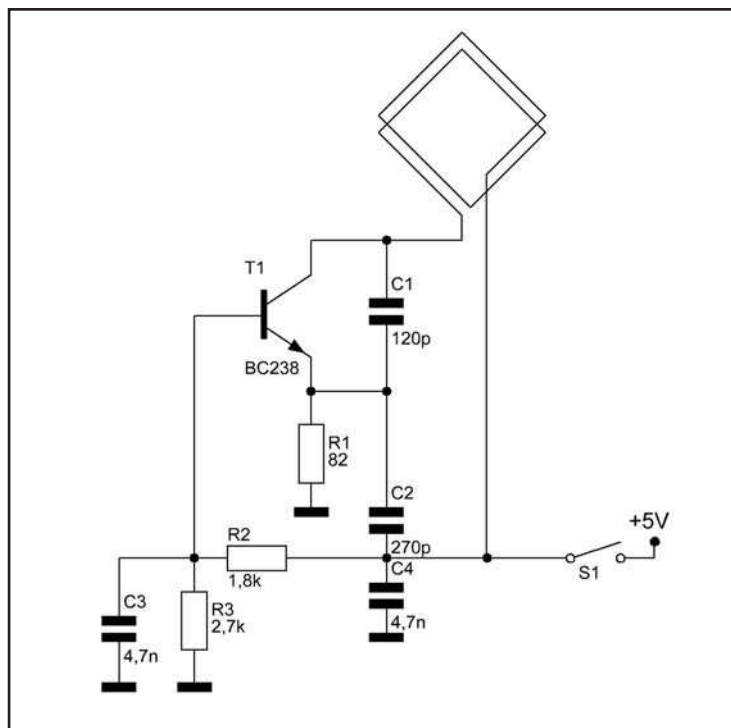
In figura è rappresentato lo schema elettrico di un oscillatore controllato dalla luce. Il circuito è un astabile a BJT dove è stato aggiunto un fototransistor denominato FPT100. Questo componente così collegato fornisce un controllo sulla corrente delle basi dei due transistor. Quando c'è assenza di luce, il fototransistor presenta un'elevata resistenza che non permette la circolazione delle correnti di base, di conseguenza i due led rimangono spenti, mentre in presenza di luce la resistenza si riduce e il lampeggiatore entra in funzione.

SPEGNIMENTO RITARDATO DEL LED

Il circuito funziona nel seguente modo: alla pressione del tasto S1, la corrente di base del transistor T1 viene deviata verso massa quindi T1 rimane interdetto, mentre entrano in conduzione T2 e T3 che fanno accendere il LED. All'apertura del tasto la corrente di base del T1 va a caricare il condensatore C1 che, una volta completata la carica manda T1 in conduzione e in interdizione T2 e T3 quindi il led si spegne. Lo spegnimento del LED dipende dal valore di C1 e dalla regolazione della corrente fatta con il potenziometro da 1K.

MISURATORE **DI CAMPO**

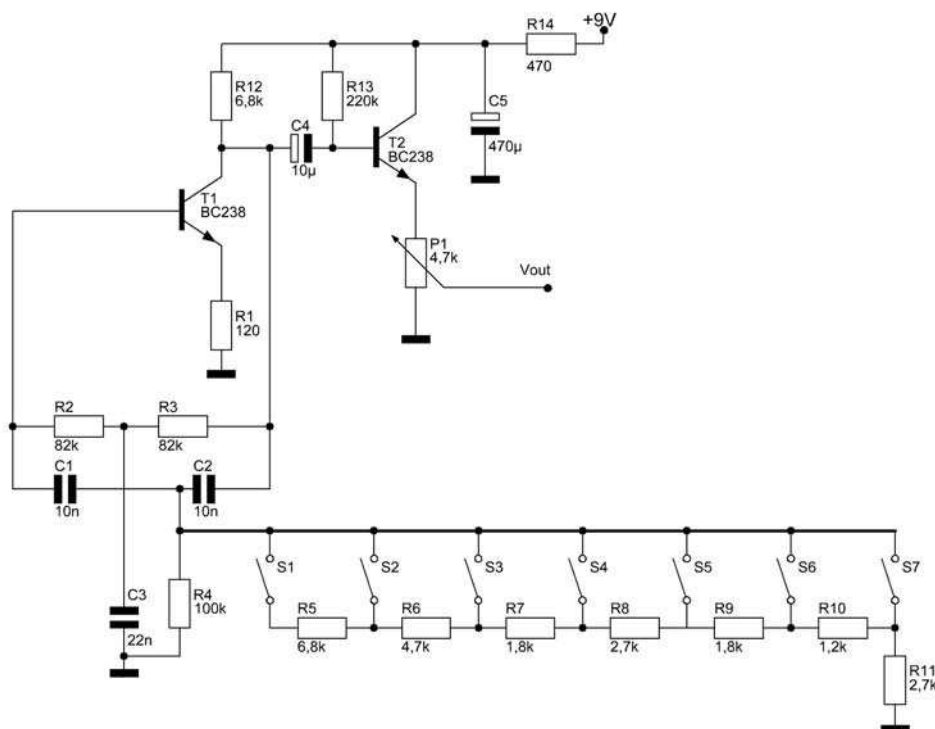
Questo semplice circuito si compone di un elemento radiante di filo smaltato con dimensioni 7,5 cm per ogni lato e da un transistor che amplifica il flusso di corrente livellato proveniente dal diodo. La bobina e il compensatore formano un circuito accordato che rilevano il segnale RF. Il valore del compensatore dipende dal tipo di segnale che si intende misurare. Il potenziometro provvede alla regolazione del milliamperometro, in modo tale che l'indicatore vada a zero in assenza di segnale. Il circuito va alimentato con una tensione di 5V ma può essere alimentato anche con una batteria da 4.5V.

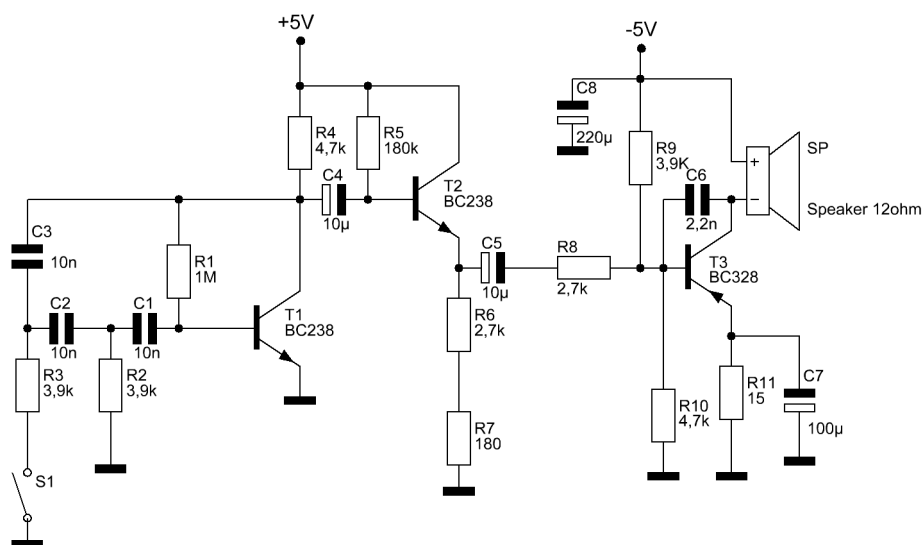
MODULO TRASMETTITORE A 27MHz

In figura lo schema di un circuito trasmettitore a 27MHz. Il circuito è molto semplice da realizzare, visto che utilizza un solo transistor NPN. L'oscillatore che genera il segnale RF è del tipo COLPITTS. L'elemento radiante è realizzato in filo smaltato è deve avere su ogni lato una dimensione di 7,5cm. Questa antenna viene chiamata in elettronica a QUADRO o a TELAIO. L'elemento radiante è parte integrante dell'oscillatore perché esso in effetti è costituito da due spire la cui induttanza risuona con la capacità dei condensatori da 120pF e 270pF. Il circuito va alimentato a 5V.

AMPLIFICATORE A BASSA FREQUENZA

L'amplificatore in questione è del tipo a simmetria complementare, ed è composto da diversi stadi. Il primo stadio è di attenuazione, per attenuare segnali aventi ampiezza elevata evitando così fenomeni di distorsione. Il segnale in uscita dell'attenuatore entra nel potenziometro e viene regolato in ampiezza per evitare che T1 venga cortocircuitato verso massa. Questo stadio viene chiamato di bootstrap ed è caratterizzato da una elevata impedenza. Il segale esce da T1 e viene diretto nel condensatore da 10uF che lo spedisce nello stadio dell'amplificatore finale. Tutto il circuito richiede un'alimentazione di 9V.

ORGANO ELETTRONICO

Lo schema illustrato è un organo elettronico costituito da un circuito oscillante e da una parte di pre-amplificazione per adattare il segnale ai circuiti a valle che possono essere, ad esempio, amplificatori finali BF. Il circuito oscillante è rappresentato dal transistor T1, che è collegato in una configurazione di tipo doppio T. Il funzionamento di questo oscillatore si basa sull'accoppiamento fra il collettore e la base del transistor per mezzo di una rete di retroazione. Ogni tasto corrisponde ad una nota. Nel momento che viene premuto, viene creato un parallelo di resistenze tra quella di valore 100K e quella che si viene a formare durante la pressione del tasto che determina una variazione di frequenza di taglio calcolabile con la formula Fo=28000/RC La parte di pre-amplificazione è rappresentata dal transistor T2. Il circuito non richiede accorgimenti particolari e viene alimentato con una tensione di 9V. .

OSCILLOFONO

In figura è riportato lo schema di un oscillofono costituito da una rete sfasatrice costituita dal transistor T1 e T2 e da un amplificatore costituito dal transistor T3. Quando il tasto risulta aperto la resistenza da 3,9K risulta interrotta, per cui la rete sfasatrice non genera più lo sfasamento di 180°, necessario perché si manifesti l'innesco delle oscillazioni. In questa situazione non si ha la produzione di alcun suono. Quando invece il tasto è premuto la resistenza da 3,9K viene collegata verso massa per cui il transistor T1 si trova nelle condizioni di generare il segnale il quale viene poi amplificato e riprodotto dall'altoparlante.

SERVIZIO MIP

MORE INFO PLEASE!

IL SERVIZIO UTILIZZABILE PER RICHIEDERE MAGGIORI INFORMAZIONI SUI CONTENUTI DI FARE ELETTRONICA

ONLINE:

www.farelettronica.com/mip

VIA FAX: 02 66508225

NUMERO DELLA RIVISTA 288

INSERISCI I CODICI MIP PER I QUALI DESIDERI RICEVERE INFORMAZIONI. UTILIZZA IL NUMERO MIP CHE COMPARE ALLA FINE DI OGNI ARTICOLO O ALL'INTERNO DELLE PAGINE DI TUO INTERESSE.

DATI DEL LETTORE

Nome ______ Cognome ______

Azienda ______ Indirizzo Azienda ______

Città ______ Cap ______ Prov. ______ Nazione ______

Tel. ______ Fax ______

e-mail ______

Dal 29 giugno al primo luglio 2009

Sesta conferenza internazionale COG 2009

Il Component Obsolescence Group (COG) annuncia la Sesta conferenza internazionale COG, 'Sfide per l'Obsolescenza e la Manutenibilità nel 21simo Secolo' che si terrà dal 29 Giugno al Primo Luglio 2009 presso il St John's College, Cambridge, Regno Unito. I due giorni di presentazioni forniranno un'analisi approfondita di una gamma di problemi di attualità legati all'obsolescenza, compresa la contraffazione, da figure di spicco dell'industria nei settori della fabbricazione, della distribuzione e dei servizi come pure dalle organizzazioni Governative: il Ministero della Difesa, UKEA e IECQ.

Dove: Cambridge (UK) **Quando:** dal 29/ Giugno a 1 Luglio 2009
Organizzazione: Component Obsolescence Group

info: www.cog.org.uk/conferences.asp

Codice MIP 2764113

26-28 GIUGNO 2009

HAM RADIO

Un appuntamento storico per i radioamatori giunto quest'anno alla sua 34° edizione. l'Ham Radio di Friedrichshafen è la fiera europea di riferimento per radioamatori ed hobbisti elettronici che attrae visitatori da tutto il mondo. In occasione della fiera si terranno workshop e conferenze di sicuro interesse per gli appassionati del settore.

***Dove:** Friedrichshafen, Germania **Quando:** 26-28 Giugno 2009*
***Orari:** 26-27/6 dalle 9.00 alle 18.00, 28/6 dalle 9.00 alle 15.00*
***Organizzazione:** Messe Friedrichshafen*

info: www.hamradio-friedrichshafen.de

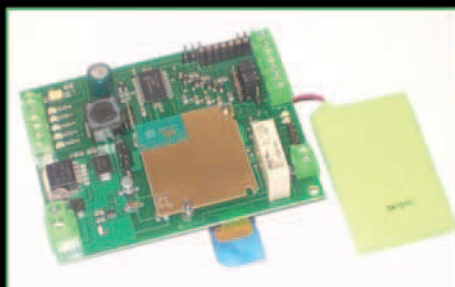
Codice MIP 2764121

4-5 LUGLIO 2009

ComputerFest & Radioamatore

Fiera dell'elettronica, CD, randiantistica, editoria specializzata, telefonia cellulare.

***Dove:** Cecina (LI) **Quando:** 4-5 Luglio 2009*
***Orari:** dalle 9.00 alle 18.00*
***Organizzazione:** Compendio Fiere info: www.compendiofiere.it*

Codice MIP 2765433

Chassis PXI

Pickering Interfaces espande la propria offerta di chassis PXI con i nuovi 40-922 e 40-923 a 8 e 19 slot PXI aventi supporto per un controller embedded a 3 slot o una interfaccia PCI-PXI. La cura nella progettazione ha conferito al sistema una alta capacità di raffreddamento pur mantenendo un basso rumore acustico grazie all'utilizzo di un sistema di controllo della velocità di rotazione della ventola, e una notevole semplicità di montaggio all'interno di un sistema rack. Lo stato della tensione di alimentazione, della temperatura operativa interna e della ventola possono essere monitorati attraverso interfaccia RS232 o grazie all'indicatore sul pannello frontale.

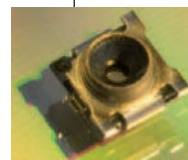
CODICE MIP 2765463

OPERAZIONALI CON CALIBRAZIONE ON-CHIP

Microchip annuncia i primi ed unici op-amp sul mercato che includono mCal, un circuito di calibrazione on-chip che calibra le tensioni di offset all'accensione, usando un rivelatore interno del power-on reset, o in base allo stato di un pin esterno. Gli op amp MCP651/2/5 in tecnologia CMOS presentano una banda di frequenze di 50 MHz e una corrente di pilotaggio massima di uscita rail-to-rail di 95 mA rendono l'MCP65x ideale anche per le applicazioni più esigenti nei mercati consumer, industriale e medicale. La valutazione del dispositivo può essere condotta per mezzo della scheda di valutazione MCP651 messa a disposizione da Microchip.

CODICE MIP 2765619

Nuovi connettori **Murata**

Murata annuncia la nuova linea di connettori RF SWG per applicazioni UWB composta da connettori di dimensioni pari a 2 x 2 x 0,9 mm che consente un collaudo affidabile dei circuiti in radiofrequenza fino alla fase finale di produzione, garantendo nel contempo una riduzione degli ingombri sulle schede estendendo la portabilità a dispositivi destinati al mercato consumer come telefoni mobili, PDA, sistemi GPS e apparati di misura. Compatibili con i più diffusi standard wireless come WiFi, WiMAX, 4G, Bluetooth, ZigBee e UWB, la famiglia si distingue per le eccellenti caratteristiche in frequenza con basse perdite di inserzione e VSWR.

CODICE MIP 2765466

DAVINCI, IL NUOVO MICROCONTROLLORE TEXAS INSTRUMENTS

Il processore TMS320DM365 di Texas Instruments basato su tecnologia DaVinci™ è stato utilizzato da VIVOTEK nelle nuove telecamere di rete IP8161 per offrire modalità night/day, compressioni H.264, MPEG-4 e MJEPG in modo da ridurre drasticamente la dimensione dei file da memorizzare pur risparmiando banda e garantendo alta qualità delle immagini e efficienza di streaming. La risoluzione di 720p HD a 30 frame al secondo permette la realizzazione di integrazioni di alto profilo minimizzando i costi per il cliente finale.

CODICE MIP 2765472

APPLICAZIONI CON CAPSENSE

La soluzione per il rilevamento tattile CapSense di Cypress basata sui dispositivi PSoC è stata utilizzata per il pilotaggio del pannello di controllo di Bamboo, il nuovo dispositivo di input grafico sviluppato da Wacom. La tavoletta Bamboo consente di inserire testi scritti a mano, disegni e annotazioni in documenti digitali supportando display a schermo ampio e offrendo una superficie di lavoro che presenta le stesse sensazioni che si hanno quando si scrive su carta. Con i dispositivi CapSense basati sulla tecnologia PSoC sono stati realizzati oltre 2,5 miliardi di tasti, cursori e altre interfacce sensibili al tocco utilizzati in sistemi prodotti in tutto il mondo.

CODICE MIP 2765470

Sistemi di videosorveglianza con registratore integrato

Registratore A/V portatile su micro SD Card

FR360

€ 99,-

Dispone di una microtelecamera CMOS a colori, un microfono, un pulsante di rec/stop, un interruttore on/off, una batteria ricaricabile (interna) agli ioni di litio e un connettore USB per lo scarico dei video e la ricarica. I file vengono salvati nel formato video 3GP (potete utilizzare Quicktime per vederli sul PC) che possono essere riprodotti anche con un telefono cellulare. Le immagini riprese dalla telecamera vengono salvate direttamente su micro SD card (max. 2 GB, inclusa) inserita nell'apposito slot. Si collega al PC tramite porta USB (2.0 / 1.1) e viene rilevato come unità di memoria esterna removibile con possibilità di trasferimento dei file con un semplice 'Copia e Incolla'. Pixel effettivi: 640 x 480, tempo di ricarica: 2 ore, consumo: 155mA, dimensioni: 97 x 25 x 12mm, peso: 32g; dimensioni file video circa 2MB per minuto.

Sensore infrarossi con telecamera e DVR

€ 249,-
CP355

È dotato di telecamera CCD pin-hole a colori, completa di microfono, e di DVR che avvia la registrazione quando il sensore rileva un movimento; l'audio e le immagini riprese dalla telecamera vengono salvate direttamente sulla memoria flash integrata, da 128MB, o su SD card (max. 1GB, non inclusa). I file possono essere trasferiti su di un PC (collegato tramite porta USB2.0/1.1) con un semplice 'Copia e Incolla'. Dispone di un'uscita A/V che permette di visualizzare su monitor o TV le immagini riprese dalla telecamera o riprodurre i video registrati e di un ingresso A/V per il collegamento di una qualsiasi sorgente di tipo video-composito. Formato video NTSC / PAL; sensore 1/4" CCD; risoluzione 420 linee TV; angolo di ripresa 120°; ottica 3,7 mm, batteria ricaricabile interna 4,2V - 1300mAh; risoluzione 25 FPS 640 x 480; sistema operativo supportato Win98, Win2000, Win XP; dimensioni 124 × 66 × 57 mm; peso 188 g. Disponibile anche nella versione completa di illuminatore IR che si attiva automaticamente al buio (cod. **CP355**).

€ 242,-
CP354

Telecomando gestione funzioni

Mediante il piccolo radiocomando fornito in dotazione è possibile controllare le funzioni start recording, stop recording, on lock, off lock, sound on e sound off.

CODICE MIP 2753543

Camera CAR a colori

NEW

Progettata per essere montata sul cruscotto di un auto, questa camera car a colori, permette di catturare sequenze video. Utile ai fini assicurativi o per puro divertimento! Dispone di un puntatore laser che permette di allineare l'obiettivo della telecamera alla strada da riprendere. Utilizzando un cavo AV è possibile collegare il dispositivo ad un monitor con ingresso videocomposito per visualizzare il video in real-time. Viene fornita con cavo USB, cavo video RCA, adattatore per auto 12 Vdc e staffa di montaggio.

CARATTERISTICHE
Registrazione su scheda SD / MMC card (non inclusa) max. 2GB; sensore d'immagine CMOS a colori; risoluzione video 320 x 240 (QVGA); angolo di ripresa 60°; formato di registrazione AVI; alimentazione a batterie o 12 Vdc tramite adattatore per presa accendisigari (incluso); consumo max. 220 mA; dimensioni 104 x 48 x 20 mm; peso 55 g; temperatura di funzionamento da 0°C a 65°C.

CAMCOLVC3

€ 99,-

Guarda il video su Youtube: http://www.youtube.com/watch?v=zcisHl1zvhU

Sistema di videosorveglianza A/V wireless con DVR

È composto da un trasmettitore radio operante sulla banda dei 2,4 GHz completo di telecamera CMOS, orientabile manualmente, e da un ricevitore A/V portatile con funzione DVR e media player, dotato di display LCD TFT da 2,5"a colori. Le immagini riprese dalla telecamera possono essere salvate (in formato MPEG4) direttamente sulla memoria flash integrata da 64 MB o su SD Card (max. 2 GB, non inclusa) che è possibile inserire nell'apposito slot. Si collega al PC tramite porta USB (2.0/1.1) e consente di trasferire i propri file con un semplice 'Copia e Incolla'. Dispone di un'uscita dalla quale è possibile prelevare il segnale A/V per inviarlo ad un monitor o TV esterno. Corredato di auricolare stereofonico può essere utilizzato per ascoltare file musicali (MP3, WAV). L'autonomia in modalità wireless/media player è di circa 3,5 ore. Cavo A/V incluso.

€ 235,-
CP347

Tutti i prezzi si intendono IVA inclusa.

di ANTONIO GIANNICO

SISTEMI real time

Cosa è un sistema Real-Time, quali sono le applicazioni di questi sistemi, le differenze rispetto ad un tradizionale sistema general purpose e quali sono, attualmente, gli RTOS proprietari ed open-source più diffusi

Il termine Real-Time (Tempo Reale) è utilizzato soprattutto in ambito informatico con riferimento a programmi e applicazioni che devono rispondere ad eventi esterni entro tempi prestabiliti, poiché solo in questo caso la risposta può essere realmente utile. La necessità che un sistema, più in generale, sia real-time nel suo comportamento, rispetto agli stimoli esterni, è intrinseca al problema e al processo stesso che si intende gestire e/o controllare. Nonostante ciò il termine è spesso riferito, più che ad un sistema nel suo complesso, come sarebbe giusto, alla sua parte informatica ed in particolare ai Sistemi Operativi Real-Time (RTOS-Real Time Operating System). Il controllo e la gestione real-time di un processo passa infatti, spesso, per l'impiego di architetture hardware dedicate e sistemi operativi e programmi applicativi appositamente concepiti. La componente hardware e quella software, in un sistema real-time, sono legate sinergicamente con il fine di ottenere ottimizzazioni temporali del processo da gestire ed elevata affidabilità. L'evoluzione delle tecnologie informatiche, d'altronde, ha permesso di riversare, gradualmente sulla progettazione software, molte problematiche tipiche di un processo real-time.

PERCHÉ REAL-TIME?

Un sistema real-time deve rispondere ad un evento entro tempi predeterminati. Questo significa che non solo l'hardware deve gestire e registrare gli eventi esterni in maniera pronta ma anche che il software deve elaborare tali dati e attuare reazioni adeguate ed efficaci che si sviluppino e si esauriscano con effetti prestabiliti entro tempi prestabiliti. Queste prime generali considerazioni permettono già di delineare quale sia, più precisamente, la differenza tra un sistema real-time ed uno non real-time. Un normale personal computer, per esempio, non è un sistema real-time: poco importa se in un certo istante si rifiuti per qualche secondo o qualche decina di secondi di rispondere ai nostri comandi perchè è impegnato in un aggiornamento, in un defrag o nel salvataggio di un file di grosse dimensioni; ci basterà attendere qualche attimo in più affinché ci restituisca realmente il controllo e risponda di nuovo in modo pronto ai nostri comandi. In altre parole la stessa operazione non sempre viene eseguita dal nostro PC negli stessi ben precisi e prestabiliti tempi a causa di molteplici fattori. Ciò non è tuttavia un problema visto l'impiego general-purpose cui il personal computer è destinato. Completamente differente è il caso di un processo industriale o di un computer specializzato per il controllo di volo di un aereo o per il controllo di un apparato bellico come può essere una postazione missilistica. Pensate cosa accadrebbe se il controllo di assetto di un aereo non rispondesse nei tempi necessari solo perché il computer stava effettuando un defrag di un disco fisso e si rendesse nuovamente disponibile quando ormai l'aereo è precipitato e la risposta non serve ormai assolutamente più a niente! La gestione di

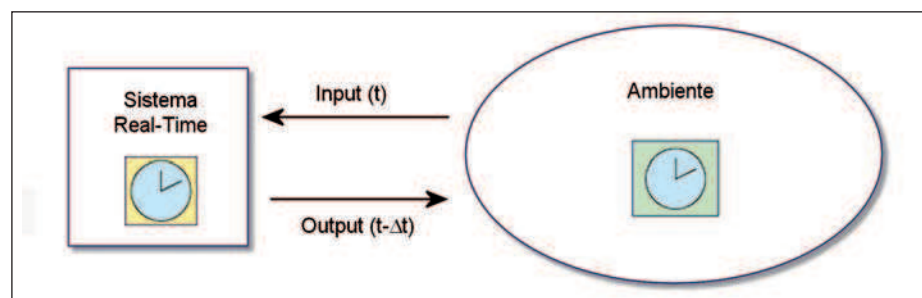



Figura 1: rappresentazione generale del concetto di Sistema Real-Time.

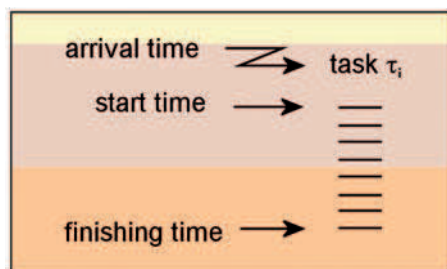
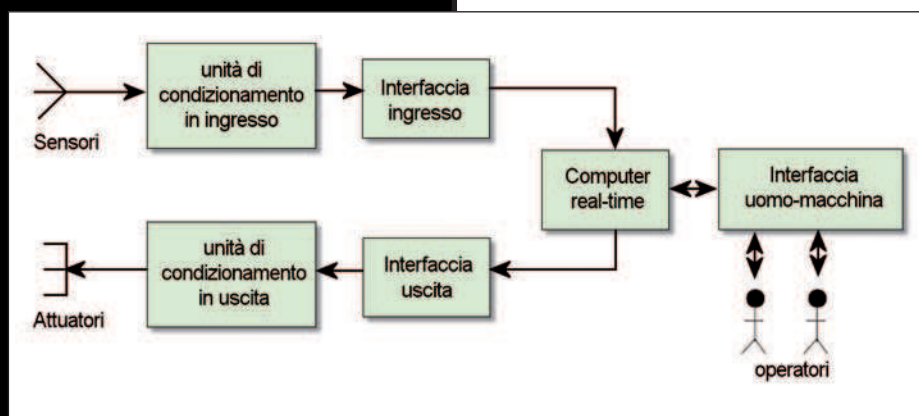



Figura 2: esecuzione di un task.

un sistema di questo tipo deve anzi (e ne capiremo meglio il significato tra breve) essere qualcosa di più che real time, deve essere "Hard-Real-Time". Solitamente la caratteristica di un sistema di essere real-time è richiesta insieme ad altre peculiarità in qualche maniera correlate. Un requisito frequentemente associato è quello della continuità operativa: alcuni sistemi real-time devono cioè lavorare in modo continuativo, per lunghi periodi, senza alcuna interruzione; è il caso particolarmente gravoso e sensibile del controllo di centrali nucleari o del controllo del traffico aereo, solo per citare due esempi. Spesso data la criticità dei sistemi real-time e le conseguenze che un malfunzionamento può comportare, compresa, a volte, la possibile perdita di vite umane, è necessario che in condizione di errore tali sistemi attivino automaticamente affidabili procedure di recupero continuando a funzionare in modalità ridotta. Per formalizzare meglio ed in modo semplice la differenza tra sistema real-time e sistema non real-time è possibile asserire, senza ambiguità, che un sistema real-time non deve soltanto eseguire le funzioni correttamente, è anche necessario che esse siano eseguite e completate nel rispetto di precisi limiti temporali. La seguente frase è certamente efficace a riassumere questo concetto: "In un sistema che deve essere real-time, la risposta giusta, se arriva in ritardo, è sbagliata!". Occorre sgomberare il campo da un possibile errore concettuale: il termine hard real-time non sottintende necessariamente criticità in termini di sicurezza. Esso potrebbe rappresentare, per esempio, semplicemente criticità per la riuscita di una determinata lavorazione: si pensi al controllo del processo di laminazione di materiali in acciaio, in cui se la laminazione non è conclusa sistematicamente entro certi tempi il materiale si indurisce vanificando la riuscita del processo, senza per questo compromettere la sicurezza degli operatori.

In ogni caso resta valido un concetto: dire in assoluto quanto velocemente un sistema debba rispondere agli stimoli dell'ambiente con cui interagisce, affinché possa essere definito real-time, non ha molto senso se non si contestualizza il tutto nella specifica applicazione. Tempi di risposta considerati real-time per una data applicazione possono non esserlo per un'altra e viceversa, motivo per cui real-time più che sinonimo di veloce deve essere inteso come sinonimo di deterministicamente predicibile.

Il fatto che la correttezza del risultato dipenda non solo dagli effettivi valori di output ma anche dal tempo in cui questi sono prodotti, significa in altri termini che il tempo di sistema deve essere strettamente sincronizzato con il tempo delle variabili ambientali **(figura 1)**. Nelle applicazioni real-time si definisce, pertanto, deadline il tempo massimo entro cui un azione deve essere messa in atto e produrre i suoi effetti per mezzo dell'attivazione di un apposito task (sequenza di operazioni a volte eseguite in maniera ripetitiva con il fine di ottenere un determinato effetto-**figura 2**). Un risultato prodotto oltre la propria deadline non è solo in ritardo, ma spesso è sostanzialmente inefficace, inutile o persino dannoso.

REAL TIME E SISTEMI EMBEDDED

In informatica il termine "embedded" (cioè incapsulato, dedicato) identifica un sistema a microprocessore progettato per una specifica applicazione, spesso caratterizzato da una piattaforma hardware ad hoc. Diversamente da un computer generico (general purpose computer), un sistema embedded è dedicato a compiti specifici ben noti già durante il suo progetto: la combinazione hardware/software è pertanto anch'essa specifica per l'applicazione e per questo motivo può essere spesso spogliata del superfluo, semplificata e ridotta al necessario favorendo altri aspetti (tipicamente l'affidabilità e

Figura 3: l'applicazione delle tecnologie embedded spazia dai lettori portatili di musica ai controlli in tempo reale di sistemi complessi quali lo Space Shuttle.

Figura 4: schematizzazione generale di un sistema Real-Time (RT).

Sensori
unità di condizionamento in ingresso
Interfaccia ingresso
Computer real-time
Interfaccia uomo-macchina
Interfaccia uscita
unità di condizionamento in uscita
Attuatori
operatori

Figura 5: classificazione semplificata degli algoritmi di schedulazione.

l'esecuzione real-time). Il software necessità spesso di essere real-time perché deve assicurare un controllo ed una gestione deterministica dei tempi di esecuzione dei task.
Un esempio di sistema embedded che è sotto gli occhi di tutti è quello delle centraline elettroniche installate sugli autoveicoli per il controllo del motore, dell'ABS e della meccanica in genere. La maggior parte dei sistemi embedded esegue ripetutamente e in tempo reale il controllo di variabili di input e attiva conseguentemente opportune azioni di output. Spesso si è portati a parlare indistintamente di sistemi real-time e di sistemi embedded, come se le due cose fossero sostanzialmente equivalenti. Sovente, di fatto, le due cose coincidono, ma soltanto perché spesso un sistema embedded è necessario che sia anche realtime, ma in generale le due cose non sono necessariamente corrispondenti. Inoltre, questo non esclude, in generale, l'impiego di RTOS su piattaforme hardware general purpose, come può essere un PC. Per gestire un processo real-time che non prevede una grossa mole di dati, al posto di un sistema embedded, potrebbe essere utilizzato anche un personal computer, riducendo al minimo il numero di programmi installati ed utilizzando un sistema operativo real-time e del software opportuno.
In ogni caso la limitatezza delle risorse fisiche richieste è relativa: alcuni sistemi embedded potrebbero anche richiedere CPU potenti, hardware dedicato ed una grande quantità di memoria per eseguire certe attività garantendo elevate prestazioni. I sistemi embedded spesso non sono dotati di hard disk; il software o il firmware si trova pertanto in memorie ROM o Flash. L'assenza di supporti di memoria che comportino parti meccaniche in movimento e l'impiego al loro posto di componenti allo stato solido, come le memorie Flash, implica maggiore affidabilità, migliori prestazioni e favorisce l'approccio real-time. Non è da sottovalutare, infatti, che i sistemi embedded devono spesso rimanere attivi per anni, ininterrottamente, senza essere soggetti ad errori; anche per questo motivo il software ed il firmware sono progettati e testati con molta più attenzione rispetto al software di un personal computer. I sistemi embedded possono essere, in casi estremi, anche fisicamente inaccessibili (come i computer delle missioni spaziali), pertanto devono essere capaci di resettarsi autonomamente in caso di perdita o corruzione dei dati, motivo per cui devono integrare circuiti e software affidabili con funzionalità di watchdog.
L'evoluzione dei sistemi embedded ad alto volume di produzione si sta spostando verso i cosiddetti "System-on-a-chip" (o SoC) che racchiudono, in un singolo circuito integrato di tipo ASIC, il microcontroller/CPU e/o il DSP, memorie, oscillatori e clock, regolatori di tensione, eventuali interfacce AD/DA e verso l'esterno (USB, ethernet, ecc…).
Non è raro che il produttore dell'hardware fornisca un BSP (Board Support Package) per semplificare il supporto e l'integrazione tra il software sviluppato ad hoc, l'ambiente operativo sottostante e l'hardware. La presenza di un Sistema Operativo in un sistema embedded non è comunque necessariamente richiesta. Su semplici microcontrollori opera spesso, ciclicamente, un singolo programma anche di poche centinaia di byte, senza alcuna sovrastruttura, in ambienti complessi possono invece trovare applicazione i medesimi sistemi operativi impiegati per scopi generali (Linux, Windows CE, ecc.) oppure in maniera più mirata sistemi operativi effettivamente real-time (come VxWorks o QNX).

APPLICAZIONI

L'interesse per lo studio dei sistemi realtime è motivato dalla diffusione crescente che essi hanno in diversi settori applicativi tecnologicamente avanzati:

- catene di montaggio, automazione industriale in genere, robotica;
- PLC (Programmable Logic Controller) e sistemi SCADA;
- controllo di processi produttivi complessi;
- controllo di impianti chimici;
- controllo di impianti nucleari;
- gestione degli scambi ferroviari;
- sistemi di segnalazione per il controllo del traffico automobilistico, ferroviario, aereo e marittimo;
- sistemi di telecomunicazione (es. telefonia cellulare, centralini telefonici);
- apparecchiature per reti informatiche come router, timeserver e firewall;
- settore automotive (centraline di controllo di motori, assistenza alla guida, ABS, iniezione elettronica, ecc…);
- sistemi spaziali;
- sistemi di controllo di applicazioni militari (es. controllo missilistico);
- sistemi di controllo di volo e navigazione automatica sugli aerei;
- strumenti di misura come oscilloscopi digitali, analizzatori logici e analizzatori di spettro;
- apparecchiature biomedicali;
- sistemi di acquisizione e monitoraggio ambientale;
- realtà virtuale;
- sistemi di stoccaggio di dati come hard disk, floppy disk o compact disc;
- elettrodomestici (forni a microonde, lavatrici, apparecchi televisivi, lettori o scrittori di DVD);
- macchine apparentemente semplici come stampanti laser, fotocopiatrici, distributori automatici e sportelli bancomat;
- console per videogiochi;
- strumenti musicali digitali quali tastiere workstation, mixer digitali e processori audio;

e questo solo per citarne alcuni.

SISTEMA REAL TIME (RTOS)

La maggior parte di noi usa quotidianamente un personal computer, quasi come fosse ormai un "elettrodomestico". Proprio dal mondo del personal computer deriva la famigliarità che la maggior parte di noi ha con il termine "Sistema Operativo". Generalmente si associa questo termine a Windows e coloro che hanno maggiore memoria storica in un caso e conoscenze tecniche software nell'altro lo associano al termine MS-DOS o Linux. Di fatto il concetto di sistema operativo nasce e si sviluppa proprio parallelamente alla tecnologia dei computer. Chi si occupa di prodotti tecnologici e soprattutto di elettronica di sistemi embedded sa bene però che il mondo dei sistemi operativi è un mondo molto variegato e che vi sono diverse categorie di sistemi operativi che si diversificano per tipologia di applicazioni (generiche o specializzate), per occupazione di risorse fisiche e per prestazioni offerte. Tra questi vi è la categoria di Sistemi Operativi in Tempo Reale o Real-Time da cui l'acronimo inglese RTOS (Real Time Operating System). Ad essere precisi è necessario fare una distinzione tra software real-time e sistemi real-time; infatti in un sistema real-time il software è solo una parte del tutto (generalmente si richiede anche un hardware specializzato). La **figura 4** mostra in modo semplificato come un sistema real-time, nel senso più generale del termine, pur essendo costituito da più unità, abbia nel Computer Real-Time e nel suo software il vero cuore pulsante. Contrariamente a quanto alcuni potrebbero pensare gli RTOS non sono necessariamente una prerogativa di computer e processori ad alte prestazioni ma anche di piccoli processori e controllori. Per questo motivo si tratta spesso di sistemi operativi specializzati e caratterizzati da piccola occupazione di memoria e limitate risorse fisiche. Inoltre, un RTOS può essere, spesso, esteso con parti scritte dall'utente. Un sistema RTOS completo di funzionalità quali interprete di comandi, stack di rete TCP/IP, file system, potrebbe essere contenuto tranquillamente anche in un floppy disk. Detto in termini semplici e puramente concettuali un Sistema Operativo Real-Time è concepito e sviluppato appositamente per supportare applicazioni software in tempo reale. Il termine tempo reale fa pensare ad un sistema che risponde in maniera immediata, tuttavia il concetto di immediato non ha molto senso se non è quantificato o riferito a qualche metro di confronto contestualizzato nella specifica applicazione. Si è già fatto cenno a come, diversamente da quello che accade nell'utilizzo di un comune personal computer, l'approccio real-time diventi spesso irrinunciabile in quasi tutte le applicazioni di controllo dei processi industriali (si pensi a catene di montaggio più o meno complesse o a sistemi industriali robotizzati) e in quelle che rientrano nell'ambito delle telecomunicazioni, ovvero in tutte quelle applicazioni in cui un ritardo che ecceda determinati tempi rischia di vanificare la finalità funzionale stessa del sistema. In un sistema operativo tradizionale, che permette il funzionamento di un comune personal computer, si può tollerare un ritardo di risposta, specie se questo permette di privilegiare la precisione dei risultati. In un sistema real-time, invece, l'esigenza primaria non è quella di avere il risultato più preciso ed accurato in assoluto, ma quello di avere il risultato della precisione sufficiente allo scopo, privilegiando il rispetto di un tempo limite di risposta, in quanto una risposta anche estremamente precisa che arrivi in ritardo diventa di fatto inefficace e quindi inutile. Lo sviluppo di applicazioni real-time è inoltre spesso critico tanto che a volte un malfunzionamento o una inaffidabilità possono generare rischi seri per cose o persone. Si pensi alla gravità che può avere un incidente in un impianto nucleare, in un volo spaziale, o nei sistemi di difesa, a causa

PER *approfondire...*

- http://wikipedia.org (Real-Time)
- http://wikipedia.org (Sistema Operativo Real-Time)
- Salvo, The RTOS that run in tiny placet-User Manual/Cap. 2-RTOS Fundamentals
- Sistemi Real-Time per il Controllo Automatico:Problemi e Nuove Soluzioni-Giorgio Buttazzo-Università di Pavia-Automazione e Strumentazione, Anno XLVIII, N. 5, pp. 107-116, Maggio 2000
- Algoritmi di schedulazione nei sistemi Real-Time FWn 15 F. Pentella

Figura 6: differenza tra la politica di schedulazione preemptive e round-robin.

di errori software che minino il comportamento real-time del sistema. Un sistema operativo ed un software real-time non devono per questo essere necessariamente più veloce di altri tipi di sistemi operativi e software; la proprietà più importante che devono possedere non è in assoluto la velocità, quanto piuttosto la deterministicità e prevedibilità della risposta nel rispetto di assegnati tempi. Prevedibilità significa capacità di determinare in anticipo, in qualunque condizione operativa, se uno o più task di calcolo potranno essere completati entro determinati vincoli temporali (deadline).

CARATTERISTICHE DI UN SISTEMA REAL-TIME

Un buon sistema real-time deve possedere le seguenti caratteristiche:

- Schedulazione ottima: tutti i task ed i vincoli temporali noti a priori e schedulazione che minimizzi al massimo la funzione di costo K_s (paragrafo successivo).
- Condivisione delle risorse: i task sono entità separate ma concorrono ad uno stesso scopo e condividono le stesse risorse fisiche.
- Garanzia di esecuzione: tutti i task di tipo hard real-time devono terminare entro le proprie deadline; nel caso in cui arrivi un nuovo task o un task non possa essere completato entro la deadline, o deve essere impedita l'esecuzione del nuovo task o deve essere successivamente recuperata l'esecuzione del task che sta per "sfondare".
- Prevedibilità delle chiamate di sistema: il sistema deve essere in grado di valutare i tempi di ogni task per determinare una schedulazione fattibile in modo da non introdurre ritardi indefiniti o inaccettabili.

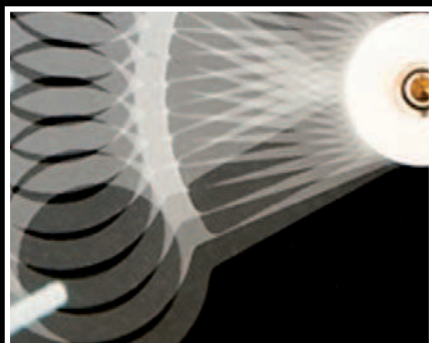
SISTEMI HARD REAL TIME E SOFT REAL TIME

Quanto esposto fin quì getta una prima linea di demarcazione tra sistemi operativi tradizionali ed RTOS. Una seconda distinzione può essere fatta all'interno dei sistemi Real-Time. Questi possono essere, infatti, suddivisi in Hard Real Time e Soft Real Time. I sistemi Hard Real Time sono detti tali in quanto il rispetto dei limiti temporali di risposta sono da considerarsi più stringenti: in altri termini un sistema di questo tipo non deve mai fallire il rispetto dei limiti temporali di risposta. Il caso tipico è quello di un processo industriale (per esempio una catena di montaggio robotizzata) in cui il mancare una certa operazione entro un certo istante può vanificare la riuscita di tutte le operazioni successive; in questo caso il danno tecnico-funzionale, oltre che economico, è assolutamente inaccettabile perché si vanifica la riuscita dell'intero processo e non solo di una parte di esso. Si pensi ancor di più al controllo della temperatura del nocciolo di una centrale nucleare: il mancato rispetto dei vincoli temporali può provocare un disastro!

Nel caso in cui la mancata riuscita di una delle operazioni del processo non implichi, necessariamente, il fallimento di fasi successive (che pertanto mantengono una certa indipendenza rispetto a quelle che le precedono), si possono tollerare sistemi operativi Soft Real Time.

Un riproduttore video o audio, per esempio, è tollerabile che sia gestito in modo Soft Real Time in quanto è necessario che la risposta sia pronta (diversamente si perderebbe la percezione di fluidità dell'immagine o del sonoro), tuttavia il non rispetto dei limiti temporali, in un generico istante t, che blocchi per una frazione di tempo la riproduzione, non pregiudica la corretta riproduzione della restante parte di filmato o di audio.

Ovviamente l'evento per essere tollerato deve essere statisticamente poco probabile.

Analogamente, un sistema di video conferenza deve visualizzare le immagini con un'alta frequenza di aggiornamento. Se fallisce frequentemente l'aggiornamento delle immagini, il sistema non può essere considerato un buon sistema

real-time, nel suo ambito, mentre se l'anomalia si presenta solo sporadicamente il sistema rimane ancora accettabile. Le temporizzazioni dei task che definiscono il controllo, la gestione e la corretta esecuzione di un processo necessitano, in un sistema real-time, di essere gestite adeguatamente; è necessario gestire l'istante di avvio di un certo task e prima ancora è necessario definire istante per istante quale task mandare in esecuzione. Questo tipo di gestione è resa possibile dal cosiddetto schedulatore il quale rende possibili le operazioni di schedulazione. Occorre sottolineare che anche hard real-time non significa in assoluto veloce in quanto certi eventi possono impiegare anche ore prima di divenire critici; Si può anche dire che in un sistema hard real time il tempo di risposta deve esser sempre rispettato in senso assoluto, mentre in un sistema soft real-time è anche accettabile che lo sia solo mediamente o comunque con elevata frequenza. In altri termini, mentre in un processo hard real-time la violazione della deadline può comportare un effetto catastrofico, in un sistema di tipo soft (alcuni sistemi di monitoraggio, sistemi multimediali, giochi di realtà virtuale, ecc…) è certamente indesiderata, ma non compromette, in generale, il corretto funzionamento del sistema, al più comporta un degrado delle sue prestazioni.

Un sistema operativo non real-time, tipo Windows, risulta inadeguato per applicazioni real-time: innanzitutto manca un isolamento temporale tra i task, per cui un picco di carico su un task può ripercuotersi sugli altri in modo incontrollato. I sistemi "hard" possono garantire la fattibilità di schedulazione di un insieme di task hard e soft real-time; i sistemi "soft" invece possono garantire la fattibilità di schedulazione di un insieme di soli task soft real-time.

Volendo formalizzare i concetti si può descrivere un sistema real-time come un sistema che, dato un insieme di n suoi task, con i relativi vincoli temporali di (deadline i-esimo), è in grado di minimizzare adeguatamente la seguente funzione costo:

$$K_s = \sum_{i}^{n} K_i(t)$$

dove Ki(t) funzione costo del task i-esimo è definita come:

$$K_i(t) = \{0(t < d_i); \infty(t > d_i)\}$$

nel caso in cui il task i-esimo è hard real-time, o come:

$$K_i(t) = \{0(t < d_i); f(t)(t > d_i)\}$$

nel caso in cui il task i-esimo è soft real-time. La funzione f(t) è monotona crescente nel tempo. Con la precedente formalizzazione si intende che i task hard real-time necessitano di una schedulazione che permetta loro di terminare in un istante t' minore della deadline in modo che la funzione costo risulti 0 e non ∞; diversamente per i task soft real-time, data la loro natura, lo sfondamento della deadline (t'>di) non genera un costo globale pari ad infinito, valore inteso come sinonimo di disastro. I sistemi operativi general purpose non sono hard real-time ma al più soft (ad esempio Linux è un sistema soft real-time).

I FATTORI CHE MINANO LA PREVEDIBILITÀ

I prodotti delle famiglie Windows e Unix non soddisfano le caratteristiche tipiche di un sistema real-time. Infatti pur gestendo l'esecuzione di più processi con pre-rilascio, non è possibile prevederne esattamente il tempo di esecuzione. Va inoltre considerato che l'impiego di hard-disk dati, dispositivi USB o altre periferiche introduce forti latenze di esecuzione da parte della CPU. Per questi motivi diventa impossibile stabilire, con certezza, quanto tempo sarà necessario per la corretta esecuzione di un task. I sistemi operativi spesso presentano caratteristiche e problematiche che aumentano la non prevedibilità dei tempi di esecuzione. Alcuni di essi sono:

- DMA: può rubare il bus alla CPU ritardando l'esecuzione di un task critico. Per questo in un sistema real-time si preferisce disattivarlo o usarlo in modalità timeslice (assegnazione in istanti prestabiliti del bus al DMA anche se non ci sono operazioni da fare);
- cache: può causare non prevedibilità poiché esistono casi in cui essa fallisce.

Dovendo considerare il caso peggiore si preferisce non usarla affatto;
- meccanismi di gestione della memoria: ad esempio la paginazione può essere causa di page fault intollerabili per un sistema hard real-time. Per questo motivo è preferibile usare la segmentazione o la partizione statica della memoria:
- le interruzioni: molti dispositivi periferici che hanno qualche informazione da scambiare con la CPU possono originare interruzioni non prevedibili che, durante l'esecuzione di un task critico, generano ritardi altrettanto non prevedibili; di conseguenza è opportuno che il loro impiego sia ridotto al minimo o che siano disattivate;
- sistemi di power management: possono rallentare la CPU o far eseguire ad essa del codice finalizzato a dissipare minor energia. Poiché in un sistema real-time è importante non sfondare una deadline piuttosto che consumare poca energia, è preferibile che questi meccanismi siano disattivati.

GESTIONE DI PROCESSI PERIODICI E DI PROCESSI APERIODICI

In un sistema real-time, la maggior parte dei processi di acquisizione, di calcolo e di controllo sono spesso di tipo periodico; quando si ha a che fare con un processo periodico si definisce periodo di esecuzione il lasso di tempo che intercorre tra due esecuzioni del task. Esistono però attività che devono essere svolte solo al verificarsi di eventi esterni, che si manifestano con cadenze irregolari e quindi classificabili come aperiodiche. In queste condizioni, vi sono due esigenze fondamentali: da un lato, è necessario rispondere all'evento esterno aperiodico nel più breve tempo possibile, dall'altro bisognerebbe non compromettere la schedulabilità dei processi periodici causando il ritardo di uno o più task oltre la propria deadline. Se le attività aperiodiche sono di tipo soft, si cerca di minimizzare il loro tempo di risposta, garantendo che tutti i task periodici (pur essendo ritardati dal servizio aperiodico) completino la loro esecuzione entro le rispettive deadline. Se alcune attività aperiodiche sono di tipo hard, si cerca di garantire off-line che la loro esecuzione termini entro le deadline specificate; ovviamente è necessario che le richieste, pur arrivando ad intervalli irregolari, non superino una frequenza massima prestabilita (in tal caso tali richieste si definiscono sporadiche). Il modo più semplice di gestire le attività aperiodiche preservando le deadline dei processi periodici è quella di schedulare le richieste aperiodiche in background. In sostanza un task aperiodico è schedulato solo quando il processore è libero, ossia quando non ci sono processi periodici in esecuzione. Ciò può comportare in ogni caso un problema: se il carico computazionale relativo ai processi periodici è elevato, il tempo residuo per l'esecuzione delle attività aperiodiche può essere insufficiente. Il tempo di risposta dei processi aperiodici può allora essere migliorato associando ad essi un processo periodico (server) dedicato appositamente al servizio delle richieste aperiodiche. Il processo è allora caratterizzato da un periodo Ts e da un tempo di esecuzione Cs. Esso è schedulato con lo stesso algoritmo utilizzato per i processi periodici e una volta attivato effettua il servizio delle richieste aperiodiche pendenti, entro i limiti della sua capacità Cs. L'ordine di servizio delle richieste aperiodiche può essere deciso in funzione del tempo di arrivo, del tempo di calcolo oppure della deadline.

LA SCHEDULAZIONE

La schedulazione definisce la politica secondo cui, in un sistema real-time, i diversi task sono messi in esecuzione. Lo schedulatore è quindi quel componente del sistema operativo che istante per istante decide quale processo o task ha diritto di detenere il controllo della CPU (time sharing). Lo schedulatore gestisce la condivisione della CPU e delle risorse di sistema da parte dei diversi processi. Esistono diversi algoritmi di schedulazione: la maggior parte di essi possono essere catalogati come di tipo peempitive oppure di tipo round-robin. L'algoritmo di schedulazione di tipo preempitive è tipicamente basato su priorità in quanto permette di sospendere temporaneamente il processo in atto a favore di un altro a maggiore priorità. Questo significa che i task sono caratterizzati da una scala di priorità assegnata e che quello in esecuzione in un certo istante deve essere sempre quello a maggiore priorità (gli altri devono essere messi in attesa in una

apposita coda). Questa politica di gestione comporta che uno o più task possono essere sospesi, temporaneamente, per lasciare risorse disponibili all'esecuzione di un task a priorità più elevata. Il modo più logico di gestire i task messi in attesa è quello di adottare una politica FIFO che permetta di recuperarli successivamente e completarli. L'assegnazione delle priorità rappresenta, ovviamente, un aspetto critico. Le priorità possono essere decise ed assegnate ai task in maniera statica, cioè prima dell'esecuzione, secondo parametri fissi, oppure possono mutare in maniera dinamica durante l'esecuzione stessa. Sicuramente una gestione dinamica rende il sistema intrinsecamente più flessibile, ma al tempo stesso più complesso. La prima osservazione che viene naturale fare è quella che in una politica di schedulazione preemptive basata su priorità le risorse potrebbero, al limite, essere occupate indefinitamente dallo stesso task. Per questo motivo potrebbe essere opportuna la gestione dinamica. In una schedulazione round-robin, invece, ad ogni task è assegnato un certo lasso di tempo di esecuzione e non sono previste priorità; al limite se si chiude il periodo di tempo assegnato ad un certo task e questo risulta ancora in esecuzione esso può essere interrotto ed il tempo di esecuzione riassegnato ad un altro task che era in coda di attesa. In altre parole un sistema con gestione round-robin è simile ad un sistema con priorità in cui tutti i task presentano lo stesso livello di priorità. In **figura 5** è riportata una classificazione semplificata degli algoritmi di schedulazione mentre in **figura 6** è riportato uno schema che evidenzia le principali differenze tra la politica di schedulazione preempitive e quella round-robin.

Chi voglia approfondire l'argomento della schedulazione può ricercare ulteriori dettagli sui seguenti algoritmi di schedulazione:

- Rate monotonic (RM): applicato a sistemi che gestiscono eventi periodici in cui è assegnata, in maniera statica, la maggiore priorità al task caratterizzato dalla maggiore frequenza temporale. Le priorità assegnate ai processi sono fissate all'inizio e rimangono invariate per tutta la durata dell'applicazione.

Un risultato importante è il seguente: un insieme {1, …, n} di n task periodici risulta schedulabile con questo metodo se (condizione sufficiente):

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq n(2^{1/n} - 1)$$

dove Ci e Ti rappresentano, rispettivamente, il tempo di esecuzione massimo e il periodo del processo i.

- Earliest deadline (ED): il task considerato, in un certo istante, a più alta priorità, è quello caratterizzato dalla più prossima deadline. Questo significa che la priorità di ciascun task non può essere statica ma varia nel tempo in modo dinamico. Ovviamente si tratta di un algoritmo preemptive in quanto, se arriva un processo con una deadline minore di quella del processo in esecuzione, quest'ultimo viene sospeso e la CPU viene assegnata al processo appena arrivato. Ciò equivale ad assegnare ad un processo una priorità inversamente proporzionale alla sua deadline. A differenza dell'algoritmo RM, ED può essere utilizzato per schedulare sia processi periodici che processi aperiodici. Un insieme {1, 2, …, n} di n processi periodici è schedulabile con ED se e solo se (condizione necessaria e sufficiente):

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1$$

- Minimum laxity first: la priorità è stabilita sulla base di un parametro detto laxity definito come la differenza tra il tempo restante al processo per raggiungere la sua deadline e quello effettivamente ancora necessario al completamento della sua esecuzione. Ovviamente questa definizione fa rientrare l'algoritmo tra quelli dinamici.

In realtà, non esiste un approccio di schedulazione statica o dinamica che possa essere considerata adeguata in senso assoluto. Ne è prova il fatto stesso che esistono algoritmi detti "a priorità mista" che tendono ad unire la gestione statica con quella dinamica. In sostanza si impiega, in generale, una politica tipo rate-monotonic ma si individuano un certo numero di task critici cui si assegna un livello di criticità da gestire opportunamente.L'argomento dello scheduling è certamente di prima importanza ma non l'unico rilevante nella descrizione di un sistema real-time: meccanismi più propriamente detti di Inter-Process Communication (IPC) cioè di comunicazione e sincronizzazione tra task sono aspetti altrettanto critici. L'efficacia della schedulazione può, certamente, ricevere beneficio dall'aumento dalla potenza computazionale del sistema (impiego di processori più potenti). Ovviamente, si potrebbe essere portati a pensare che non valga la pena studiare e progettare tecniche apposite per sistemi RT, poiché i computer sono ormai sempre più veloci. In realtà una affermazione del genere non è propriamente corretta, in quanto anche la complessità dei processi può aumentare ed in ogni caso aumento di velocità non significa certezza che i vincoli temporali di ogni attività siano rispettati. La riduzione dei tempi di esecuzione, inoltre, non può non passare anche per l'ottimizzazione del codice applicativo. Infatti non è raro che l'ottimizzazione anche di piccole parti critiche del codice comporti una riduzione percentualmente rilevante dei tempi di esecuzione di task critici.

SCELTA DI UN RTOS

Gli aspetti che portano a scegliere un sistema operativo RTOS piuttosto che un altro sono molteplici. Esistono diversi sistemi operativi RTOS, sia commerciali che open-source. L'analisi dell'applicazione, l'adeguatezza rispetto all'applicazione stessa del metodo di schedulazione, la gestione e l'occupazione delle risorse fisiche del sistema (l'occupazione di memoria in particolare, visto che spesso un sistema dedicato dispone di poca memoria), la disponibilità di documentazione e supporto tecnico sono tutti fattori alla base della scelta di un RTOS. Oltre a questi aspetti è importante, per chi sviluppa applicazioni per sistemi real-time, disporre di strumenti adeguati per effettuare operazioni di debug. Per questo motivo, i produttori di questi sistemi operativi, generalmente, non forniscono solo il kernel ma anche strumenti integrati di compilazione e di debug, spesso scalabili in termini di moduli e librerie, che si possono integrarare a seconda delle specifiche esigenze. Un importante requisito

che un Sistema Operativo Real Time dovrebbe avere è infatti proprio la scalabilità dal momento che in molte applicazioni non è necessario disporre di tutte le funzioni. In questo modo si evita di appesantire inutilmente il sistema rendendo insufficienti le risorse (prima tra tutte la memoria). Una discussione a parte andrebbe inoltre aperta a riguardo della portabilità del codice. In generale la scelta di un Sistema Operativo Real Time deve tener conto di:

- consolidate caratteristiche di robustezza ed affidabilità;
- scalabilità e modularità che permettano di alleggerirlo nelle parti inutili per la specifica applicazione e possibilità, eventualmente, di essere modificabile nei sorgenti per essere adattato a situazioni applicative specifiche (open source);
- porting su diverse CPU;
- già affermato nell'ambito delle applicazioni real-time;
- aperto alla gestione ed all'integrazione del maggior numero di protocolli di comunicazione possibile;
- rispondenza a certificazioni IEEE specifiche (POSIX-Portable Operative System Interface).

Tra gli RTOS commerciali troviamo infatti i POSIX-conformant (ad esempio LynxOS che è Unix compatibile) e non POSIX-conformant come ad esempio VxWorks (che supporta in parte gli standard POSIX).

La scelta di un sistema operativo Real Time può essere fatta sia rivolgendo l'attenzione verso RTOS proprietari che verso RTOS a tutti gli effetti Open Source. Ovviamente, un sistema operativo opensource presenta il vantaggio competitivo legato all'economicità.

Inoltre essendo un software libero, è disponibile anche nel codice sorgente e per questo modificabile ed adattabile alle specifiche esigenze eliminando ciò che è eccessivo rispetto alla propria applicazione: ciò incrementa e rafforzare la robustezza dell'applicazione senza aggiunta di costi particolari.

Di conseguenza il software modificato può a sua volta essere liberamente distribuito, anche in questo caso senza aggravi particolari di costi. Generalmente non è difficile scaricare liberamente questi sistemi operativi dal web.

PRINCIPALI RTOS PROPRIETARI

ITRON-TRON

The Real-Time Operating System Nucleus è un RTOS il cui progetto fu iniziato nel 1984 dal Prof. Dr. Ken Sakamura (università di Tokyo) e portato avanti con successo tanto che la versione ITRON (Industrial TRON) è diventata uno dei sistemi operativi maggiormente incorporato nei dispositivi elettronici di produzione giapponese. Vi sono diversi sistemi derivati come JTRON (Java TRON che permettere l'uso di una piattaforma Java), BTRON (Businnes TRON), CTRON (Central and Communications TRON pensato per mainframes e sistemi di telecomunicazione, switch di rete, bridge) rappresentante lo standard delle compagnie di telecomincazione giapponesi, MTRON (Macro TRON, per interconnessioni tra differenti componenti TRON). Alcune note storiche: nel 1989 Matsushita introdusse il primo BTRON PC basato su 80286 INTEL a 8MHz, 2 Mbyte di memoria e dual boot per consentire la scelta tra sistema BTRON ed MS-DOS.

Il governo giapponese intendeva installare il sistema BTRON PC sulle macchine delle scuole. Il governo degli Stati Uniti vide in questa iniziativa una concreta minaccia per il proprio mercato e intimò ritorsioni. Ciò frenò le intenzioni dei giapponesi in parte dipendenti dagli americani nelle esportazioni. Nonostante tutto ITRON è sopravvissuto incorporato nell'elettronica per auto, nella robotica, nell'elettronica di satelliti e nell'automazione industriale, specie quella di produzione cinese.

QNX-RTOS

Unix-like POSIX-compliant commerciale, basato su un microkernel per sistemi embedded, concepito come un insieme di piccole parti (ciascuna delle quali offre uno specifico servizio) denominate "server" che prendono il posto dei "device driver" dei sistemi operativi tradizionali. Questo aspetto costituisce la differenza principale tra un sistema microkernel ed il kernel di un sistema operativo tradizionale. Un sistema operativo tradizionale, infatti, è composto di molti moduli raggruppati in un unico grosso software (kernel monolitico) che non

possono essere separati dal resto. Un sistema microkernel come QNX invece, consente di assemblare il sistema operativo utilizzando solo le parti realmente necessarie e tralasciando il resto senza modificare o ricompilare il sistema operativo stesso. QNX risale ad un progetto di Gordon Bell e Dan Dodge i quali convinti della possibilità di sfruttare commercialmente un kernel RTOS, fondarono nel 1982 la Quantum Software Systems nell'area high-tech fuori Ottawa. Nello stesso anno rilasciarono la prima versione di QNX per architetture Intel 8088. Alla fine degli anni '80 il kernel fu riscritto secondo lo standard POSIX e chiamato QNX4. Grazie all'interfaccia POSIX, il passaggio di software Unix su QNX risultò molto agevole. Verso la fine degli anni '90 QNX fu riprogettato come sistema operativo per il supporto di sistemi multiprocessori prendendo il nome QNX Neutrino (2001). QNX era dotato in origine di licenza per sviluppi non commerciali (QNX NC), licenza sospesa nel 2003; attualmente è possibile scaricare il sistema con licenza limitata a 30 giorni. Il sistema di sviluppo si avvale della piattaforma Eclipse sviluppato in java, motivo per cui può girare virtualmente su qualsiasi sistema dotato di java e di una interfaccia grafica compatibile, tra cui Windows, sistemi Unix, Linux e QNX stesso. Grazie alla struttura microkernel QNX può diventare anche un sistema operativo distribuito (distributed operating system). La tecnologia sviluppata da Dan Dodge e Peter Van der Veen permette di creare reti di sistemi QNX che possono formare un supercomputer e gestire in parallelo più applicazioni su macchine diverse. Nel 2004 la società fu venduta al gruppo Harman International Industries e nel 2007 è stato rilasciato il codice sorgente di QNX sotto licenza open source ibrida. QNX è stato molto utilizzato soprattutto in ambiente Automotive.

RT-11

(RT-Real Time) è un RTOS monoutente non multitasking che fu progettato per la famiglia di computer PDP-11 della DEC che poterva essere contenuto su un semplice floppy. Il sistema consentiva di svolgere la maggior parte dei compiti di un comune personal computer, eccetto la navigazione sul web.
Uno dei fattori che consentiva maggiore velocità dell'RT-11 era il file system; infatti tutti i file erano contigui e ciascun file occupava blocchi consecutivi sul disco (dimensione del blocco: 512 bytes); un file poteva di conseguenza essere scritto/letto molto velocemente. Gli sviluppi che seguirono le prime versioni portò DEC ad introdurre l'RSX-11, un sistema multitasking; RT-11 rimase in ogni caso per molti un riferimento per i sistemi di acquisizione dati in tempo reale. Non ci addentriamo ulteriormente nella descrizione delle sue caratteristiche dal momento che è un RTOS ormai non più sviluppato.

VRTX

Versatile Real-Time Executive è un RTOS hard real-time con supporto al multitasking orientato ai sistemi embedded e SoC il cui core è un nanokernel sviluppato per girare su hardware minimi (un microprocessore e della RAM) di diverse architetture (ARM, MIPS, PowerPC e altri microprocessori RISC). Nato nei i primi anni 80, per merito di James Ready e Colin Hunter presenta a tutt'oggi un supporto minimo sotto il marchio Microtec. Esistono più versoni di VRTX: VTRXsa e VRTX32 sono due esempi. Quest'ultimo permette il rescheduling solo dopo il termine di una chiamata di sistema. VRTXsa invece può interrompere i task e avviare la procedura di rescheduling anche durante l'esecuzione di una chiamata di sistema, ovvero quando un task è in esecuzione in kernel mode. Per i curiosi una nota: VRTX viene utilizzato per gestire il Telescopio Hubble.

VxWorks

RTOS Unix-like della Wind River Systems (Alameda-California, USA) generalmente usato per sistemi embedded. Include un kernel multitasking, uno scheduler di tipo preemptive, gestione rapida degli interrupt, inter-process communication, funzionalità di sincronizzazione e un file system.

mipOS

(Multum In Parvo Operating System, cioè letteralmente “molto in poco”)- RTOS per microcontrollori con limitate risorse (soprattutto di memoria) che nasce parallelamente alla stesura dell’articolo “Realizzare un sistema multitasking in C”-Computer Programming N.149 - Settembre 2005, autore A. Calderone. Il suo impiego è rivolto principalmente ai i system on chip (SoC) nell’elettronica industriale e di consumo. Il firmware per questi sistemi è infatti normalmente monolitico (senza un sistema operativo vero e proprio) cioè un codice con un loop principale ed una serie di routine che formano, nel complesso, una macchina a stati. Un sistema operativo come mipOS, pensato per applicazioni embedded, permette invece la realizzazione di firmware più robusto e strutturato, con maggiore modularità e portabilità. MipOS appare adatto alle applicazioni real-time, soprattutto nell’ambito dei sistemi di controllo. La portabilità è garantita dal fatto che è quasi interamente scritto in linguaggio C. La licenza è gratuita per qualunque applicazione didattica non commerciale. Qualunque altro utilizzo deve essere concordato con l’autore.

S.Ha.R.K.

Soft Hard Real-Time Kernel. kernel Posix Standard completamente configurabile progettato sia per sistemi hard e soft real time che per sistemi non real-time e caratterizzato da una certa flessibilità di modifica della politica di schedulazione.

PRINCIPALI RTOS OPEN-SOURCE

BeRTOS

RTOS open source adatto ad applicazioni embedded che gira su molti processori e microcontrollori, da quelli a 8 bits fino a quelli più potenti a 32 bit. La licenza è di tipo GPL con una particolare eccezione che permette al codice applicativo proprietario di rimanere closed source. Le sua struttura modulare gli consente di essere utilizzato su architetture che spaziano dai piccoli microcontrollori a 8 bit (es. Atmel AVR) alle architetture ARM fino all’emulazione su Linux o Windows per scopi di debug.Oltre al kernel multitasking dispone di driver generici per il controllo di diverse periferiche hardware (seriali, ADC, display LCD, tastiere, memorie, ecc...)

ChibiOS/RT

RTOS per applicazioni embedded compatto ed efficiente dotato di licenza GPL3. Il Kernel supporta il multithreading, timer software, I/O sincrono o asincrono con possibilità di timeout. E’ supportato da diverse architetture (Intel 80386, ARM7, ARM Cortex-M3, Atmel AVR, Texas Instruments MSP430). Il kernel può funzionare anche in un processo Win32 in una modalità di emulazione I/O il che consente di sviluppare applicazioni senza dover utilizzare alcun tipo di hardware.

Chorus

kernel presente sia in versione multiprocessore che in versione commerciale, molto usato per lo sviluppo di sistemi embedded nell’ambito delle telecomunicazioni e caratterizzato da una notevole scalabilità. Il codice sorgente è disponibile sotto la licenza Sun Public License Lite Version 1.0, la documentazione è invece disponibile su docs.sun.com.

eCos

RTOS open source free, real-time per applicazioni embedded caratterizzato da un mini Real Time Kernel di cui è possibile effettuare il download e che può essere usato in applicazioni commerciali.

Haiku

Chiamato OpenBeOS nel 2001 e poi mutato in HaiKu per evitare possibili problemi legali con Palm Inc., che aveva acquistato i diritti di BeOS, può essere considerato derivante da quest’ultimo. La compatibilità con BeOS facilita il passaggio verso Haiku. Questo consente agli utenti di Haiku di avere programmi già funzionanti e testati potendo riutilizzare le applicazioni già scritte per BeOS.
Attualmente alcune parti sono considerate complete, altre sono nella fase di beta o di sviluppo.

TinyOS

RTOS open-source progettato per applicazioni wireless embedded, pensato per minimizzare le dimensioni del codice, come la memoria di molte applicazioni embedded richiede. Il suo sviluppo è stato condotto da un consorzio guidato dall’Università della California - Berkeley in collaborazione con Intel Research. Il nucleo (kernel) ha uno scheduling a due livelli gerarchici: uno per gli eventi e uno per i task. Gli eventi hanno diritto di prelazione reciproco e sui task. L’esecuzione dei task segue una politica First-Come First-Served tradizionale, senza diritto di prelazione (non-preemptive). Tale modello di concorrenza funziona bene per applicazioni con prevalenza di operazioni di I/O ma può generare difficoltà in applicazioni con prevalenza di elaborazioni (applicazioni CPU bound).

CONCLUSIONI

Come si è avuto modo di comprendere, l’argomento dei sistemi Real-Time ed in particolare dei sistemi operativi RTOS è piuttosto ampio e articolato. Questi ultimi molto spesso sono dei microkernel. Un confronto diretto tra sistemi microkernel e sistemi a kernel monolitico indica che i sistemi microkernel non hanno, in generale, le stesse prestazioni dei sistemi a kernel monolitico.
Ovviamente, la maggior efficienza di un kernel monolitico è generalmente pagata in termini di affidabilità. Un crash in un servizio di un micro kernel, infatti, causa, di norma, solo la perdita di quel servizio e non dell’intero sistema mentre in un kernel monolitico spesso il crash è generale.
In ogni caso i sistemi operativi real-time non sono pensati per sistemi desktop, piuttosto per sistemi specializzati single-purpose.
Questi, specie se si tratta di sistemi hard real-time sono progettati tenendo conto dei sovraccarichi, della tolleranza ai guasti e quindi secondo assunzioni tipicamente pessimistiche. In generale non è comunque detto che un sistema dedicato debba necessariamente adottare un RTOS tanto che molte applicazioni commerciali si basano su implementazioni tradizionali di tipo ciclico per eventi. La scelta di un sistema operativo Real Time piuttosto che di uno general-purpose dipende fortemente dall’uso che se ne deve fare, cioè dalla specifica applicazione e in misura considerevole dalla quantità di memoria e dalle risorse fisiche disponibili. Data la complessità e l’ampiezza dell’argomento si consiglia, a chi voglia approfondirne i concetti, di accedere direttamente agli specifici siti web degli RTOS illustrati nei precedenti paragrafi. ❏

CODICE MIP 2756755

di ROBERTO PRESTIANNI

Controllo Radio per motori DC *(parte terza)*

il FIRMWARE

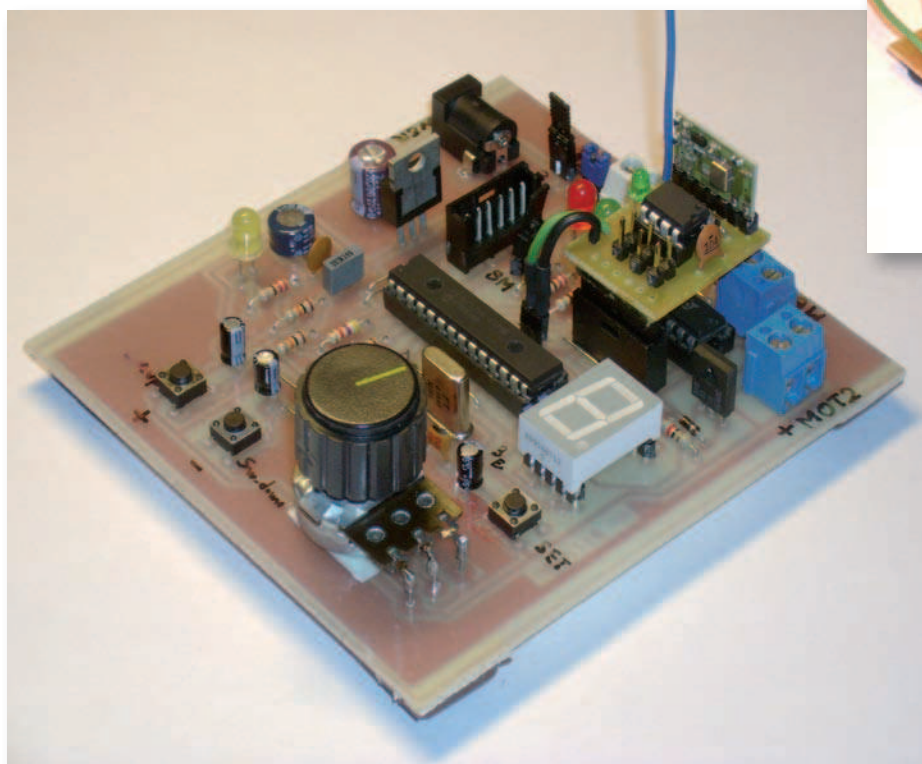
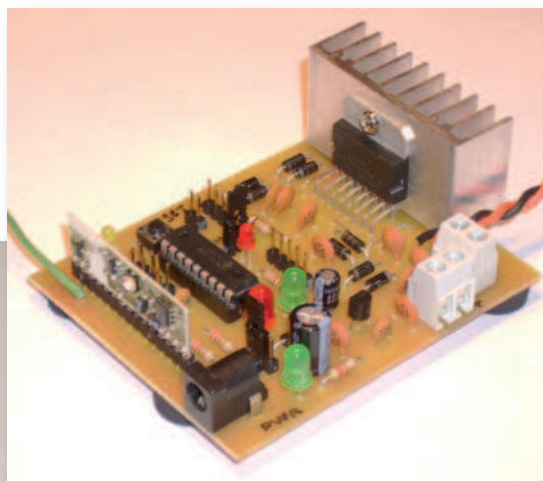
Realizziamo un sistema di controllo a radiofrequenza per motori in corrente continua: espandendo il progetto del 'Driver per motori DC', costruendo un trasmettitore ed un ricevitore RF appositamente progettati, avremo la possibilità di pilotare a distanza e senza fili un motore DC

Nel numero precedente della rivista ci siamo occupati della realizzazione del ricevitore e del collaudo dell'intero sistema radio. Adesso entriamo nel merito della logica di funzionamento del progetto, analizzando il firmware che gira nei tre diversi microcontrollori.

DESCRIZIONE GENERALE

Prima di affrontare l'argomento vero e proprio di questa ultima puntata, riassumiamo brevemente la descrizione del nostro sistema, così da facilitare la comprensione di quanto seguirà. Disponiamo dunque di una scheda base, il progetto del 'Driver per motori DC' esaminato in passato e qui modificato sia a livello software (firmware del microcontrollore PIC16F876A), sia a livello hardware con l'inserimento di un piccolo modulo trasmettitore anch'esso basato su microcontrollore. Quest'ultimo è un PIC12F675 ed è accoppiato ad un piccolo modulo radio a 433,92MHz, necessario alla trasmissione senza fili dei parametri di funzionamento del motore DC remoto. Ultimo componente del progetto è il ricevitore il quale, oltre ad intercettare ed interpretare i dati seriali trasmessi dalla sua controparte (il Tx), pilota direttamente il motore. In altre parole e più dettagliatamente abbiamo l'utente che, per mezzo di pulsanti, manopola e interruttori presenti sulla scheda base, controlla la ve-

locità e il verso di rotazione dei motori, lo step di incremento potenza eccetera, poi la scheda stessa che interpreta i comandi, modifica i parametri di funzionamento e li passa al modulo trasmittente. Quest'ultimo (Tx) legge queste informazioni passatigli serialmente secondo una precisa formattazione in nibble e, a sua volta, li mette in aria via radio anteponendo all'informazione utile un codice identificativo da 12 bit. Il ricevitore (Rx), se posto ad una distanza non superiore alla portata del sistema radio, rileva i dati in aria e li interpreta: si occupa del riconoscimento del codice ID del Tx, modifica il D.C. del-

l'onda PWM e il verso di rotazione del motore, e provvede al suo arresto di emergenza quando richiesto.

SCHEDA BASE

Il firmware della scheda base è stato modificato onde garantire la comunicazione seriale con il Tx, per mezzo della linea INT/RB0 del microcontrollore U2. Per questa funzionalità aggiuntiva è stata inserita nel project la funzione *transmit_serial_data(byte1,byte2)*, che consente proprio questa comunicazione seriale. Si fà presente che i tre projects, sviluppati in ambiente MPLAB, sono stati compilati con il PICC della Hitech software versione 9.60. Tuttavia, vista la semplicità dei codici e la facile comprensione di questi, non sarà difficile adattare i codici sorgente ad altri compilatori. Prima di analizzare il flowchart della transmit_serial_data(byte1,byte2), è bene ricordare che la trasmissione seriale, sulla quale si basa il nostro sistema, può essere disabilitata in qualsiasi momento per mezzo di una semplice procedura, così come già spiegato in un articolo precedente. Adesso esaminiamo il flowchart di **figura 1** e, per meglio comprendere il funzionamento della routine, accompagniamo la sua descrizione con la **figura 2** che ripropone una tipica comunicazione seriale tra scheda base e Tx. Dopo le necessarie inizializzazioni delle variabili locali, troviamo la generazione dell'impulso si Start-Transmission (sequenza in rosso di figura 2) che, come sappiamo, dà il via alla trasmissione seriale dei dati (12 bit utili + un nibble per la verifica sui bit). Il primo dato che viene trasmesso è il contenuto del registro CCPR1L, ovvero il byte alto del dato da 10 bit che determina il D.C. dell'onda PWM. Questo byte è ovviamente "intermezzato" da un impulso di sincronizzazione, per i motivi spiegati nel primo articolo. Segue l'ultimo nibble informativo contenente gli ultimi 2 bit del dato PWM (CCP1X e CCP1Y) e i bit contenenti l'informazione sul verso di rotazione (rot_an e rot_or). Per concludere, il nibble di conteggio del numero di stati logici '1' contenuti nel dato appena emesso dal pin RB0 di U2. Per quanto riguarda la

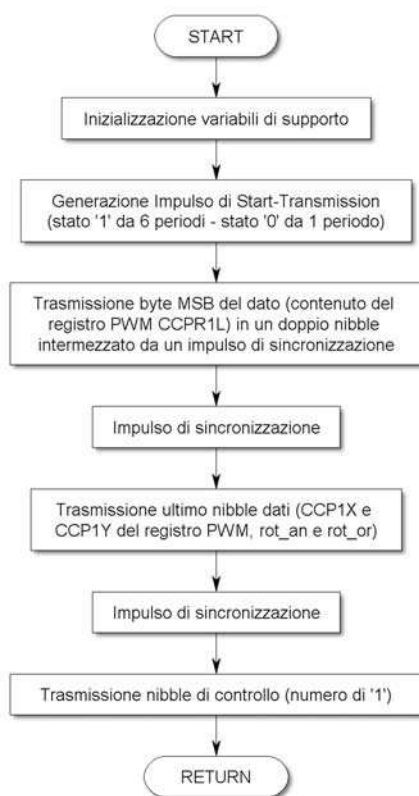



Figura 1: flowchart della funzione transmit_serial_data(byte1,byte2).

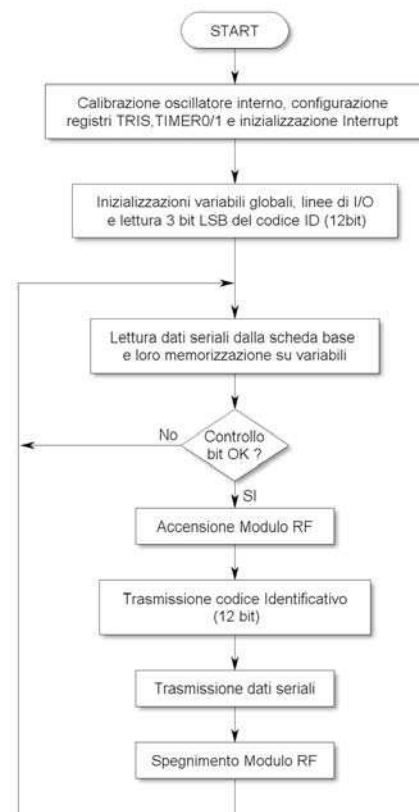



Figura 3: flowchart del firmware del Trasmettitore.

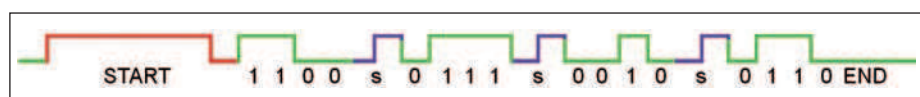



Figura 2: esempio di comunicazione seriale tra scheda base e Tx.

scheda base, possiamo concludere qui evidenziando solo alcuni punti del codice sorgente originale (quello del Driver per motori DC), che supportano la funzione di trasmissione seriale. In questi punti vi è stata fatta una modifica o semplicemente vi è stata aggiunta una nuova linea di codice. Vediamone qualche esempio tra i più significativi iniziando col **listato 1**, il quale riporta una porzione di codice che potrete trovare alla fine della *main()*: qui viene confrontato il livello di potenza appena calcolato (richiesto dall'utente), con quello precedentemente elaborato (power_l_store). Nel caso queste due quantità fossero diverse, si procederà con la trasmissione del nuovo livello di potenza desiderato e l'aggiornamento del D.C. dell'onda PWM locale. Questo accorgimento fa si che si trasmetta solo quando vi è una reale necessità, ovvero solamente quando l'utente abbia richiesto una modifica della velocità di rotazione del motore. Osservate inoltre la presenza del test sul flag "transmit_en_flag", il quale informa della abilitazione della funzionalità di controllo via radio. Di seguito la chiamata della routine appena esaminata (*transmit_serial_data(pow_byteH,pow_byteL)*) che consegna alla routine le variabili *pow_byteH* e *pow_byteL*, contenenti le informazioni utili opportunamente formattate. Il **listato 2** invece, è tratto dalla funzione *write_inc_disp()* che consente il settaggio di alcune impostazioni, come già visto in passato per il "Driver per motori DC": qui viene abilitata la funzione di trasmissione seriale, qualora si intenda ser-

virsene. Questo codice modifica il flag transmit_en_flag e lo memorizza permanentemente sulla EEPROM Dati, così come avviene per il Display.

TRASMETTITORE

Prima di esaminarne il firmware, ricordiamo che il modulo Tx deve per prima cosa leggere i dati seriali provenienti dalla scheda base, e poi ritrasmettere il dato utile anteponendo il codice identificativo. Iniziamo col descrivere la parte inziale del codice sorgente che ritroviamo stilato nel file *Tx_main.c.* Questo codice in C riporta la *main()* e tutte le funzioni si supporto: *set_freq()* per la taratura della frequenza di campionamento del segnale seriale di ingresso, *read_serial_data()* per la lettura dei dati seriali, *transmit_codeID()* per la generazione del codice identificativo del Tx, transmit_serial_data() per la trasmissione dei dati essenziali al controllo del motore (12 bit informativi + nibble di verifica bit), e poi tutte le funzioni generanti ritardi come la *Tx_period()* che determina la cadenza di trasmissione e ricezione dei singoli bit seriali, la Delay100Us per ritardi multipli di 0,1sec e *DelayMs()* e *DelayMss()* per ritardi multipli del msec. Il codice inizia con l'inclusione dei file di libreria (.h), della configuration bits ed in particolare dell'abilitazione dell'oscillatore interno da 4MHz, della dichiarazione delle variabili globali, dell'array da 12 elementi per la memorizzazione del codice identificativo, e della definizione dei nomi mnemonici delle 6 linee di I/O del microcontrollore. Il flowchart è riportato in **figura 3**, mentre la **figura 4** ripropone una tipica sequenza binaria emessa dal Tx verso il modulo RF. Il sistema si inizializza calibrando l'oscillatore interno e configurando i registri fondamentali quali il TRIS del port di I/O, i due timer e gli interrupt. Fanno seguito le inizializzazioni di alcune variabili globali e la lettura delle tre linee di ingresso GP0/1/2, ovvero i 3 bit che personalizzano il codice identificativo (codeID2/1/0). A questo punto il micro si mette in attesa della sequenza di Start-Transmission e a seguire della sequenza binaria contenente l'informazione. A questo provvede la funzione *read_serial_data()*. Andando a leggere il codice sorgente osservate come inizialmente si intenda riconoscere lo Start-Transmission e, per fare ciò, si attende inizialmente un fronte di salita, dopodiché si contano con cadenza di *Tx_period()* il numero di stati logici '1'; se si giunge al conteggio di 5 stati alti consecutivi il riconoscimento si può ritenere

LISTATO 1

```
//MODIFICA per il trasmettitore <——————————————————————————————————
//Se power_l Ë stata modificata...............
if(power_l_store!=power_l)
{
   /***** Divisione  di power_l (10 bit) in un ottuplo bit MSB
     (da assegnare a CCPR1L), e 2 bit LSB (per CCP1X e CCP1Y) *****/
   div_2byte(power_l);
   //PWM duty cycle = (CCPR1L:CCP1CON<5:4>)ïTOSCï(TMR2 prescale value)
     CCPR1L=byteH;          //Byte alto
     CCP1X=B1;CCP1Y=B0;     //Ultimi due bit meno significativi
   pow_byteH=byteH;  //<————————————————————————————————
   pow_byteL=128*B1+64*B0+32*rot_an+16*rot_or;//<——————————————————————-
   //Trasmissione seriale dei dati di potenza (PWM(power_l=ByteH+B1+B0),rot_an & rot_or) <—
—
   if(transmit_en_flag)
   {
     transmit_serial_data(pow_byteH,pow_byteL);
     DelayMs(50);           //Attesa fine trasmissione dati verso il ricevitore
   }
}
power_l_store=power_l;              //Salvataggio ultimo valore di potenza
//*******************************************************************************************
***
```

concluso. A quel punto si attende il fronte di discesa e si continua con la lettura dei bit dati in coda. Qualora il conteggio non raggiunga 5, a causa di un ritorno allo stato basso, il firmware ricomincia daccapo in attesa di uno Start-Transmission valido. Osservate inoltre come avviene la lettura del singolo nibble ed il riconoscimento dell'impulso di risincronizzazione. Dopo la lettura di una eventuale sequenza valida, viene eseguito il controllo sui bit: questo viene eseguito confrontando il numero di stati logici '1' letti durante il campionamento del dato utile, col nibble di conteggio di '1' passato direttamente dalla scheda base. Se le due quantità risultano coincidenti allora la lettura viene ritenuta assolutamente valida e si dà inizo alla trasmissione seriale verso il modulo RF. In caso contrario il PIC si mette di nuovo in attesa di una sequenza valida. Ecco allora che viene alimentato il modulo RF (contemporaneamente al led di segnalazione D1) e viene chiamata la funzione *transmit_code_ID()*. Questa, prima di emettere i 12 bit del codice identificativo genera, come di consueto, lo Start-Trasmission (vedere **figura 4**). Il modo di operare di questa funzione è il seguente: dopo la generazione dello Start-Transmission viene poi emesso il codice da 12bit, prememorizzato su di un array da altrettanti elementi (codeID[counter]). Anche questo è formattato in nibble intermezzati da impulsi di sincronizzazione, formattazione realizzata con 3 cicli for(); in ciascuno dei loop interni dei for() viene puntato l'elemento dell'array seguente ed emesso verso il modulo RF. Adesso è la volta dei 12 bit dell'informazione vera e propria, e a questa funzione provvede la *transmit_serial_data()*. Osservando le definizioni di queste due ultime funzioni, ci si rende conto che l'intera sequenza viene spezzata in gruppi da 4 bit (nibble), ognuno di essi preceduti da un impulso di sincronizzazione. Una volta trasmesso i dati, sia il modulo RF che il led vengono spenti per poi ricominciare da una nuova lettura.

RICEVITORE

Il codice stilato per il ricevitore è sicuramente un po' più complesso dei primi due visti per il lato trasmissione, questo perché deve provvedere, oltre che alla ricezione delle sequenze binarie, alla generazione dell'onda PWM a Duty Cycle variabile, e quindi ai controlli sulla potenza di uscita.

Il progetto prevede l'inclusione di un solo file in C nominato *Rx_Driver_MotDC.c*,

LISTATO 2

```
//************** Aggiunta  <————————————————————————————————————————————————————
//Attivazione/disattivazione Trasmissione dati di potenza via seriale
if(sw_up)
{
  DISP=chr_Tx;                          //Stampa carattere di 'Trasmissione seriale'
  DelayMs(250);DelayMs(250);DelayMs(250);  //Ritardo
  //Trasmissione seriale disattivata
  if(transmit_en_flag==1)
  {
  transmit_en_flag =0;                  //Disabilitazione Trasmissione RF
  transmit_serial_data(0,0);            //Annullamento potenza in RX (NOTA:deve essere e-
seguita prima
  //dell'attivazione del TIMER1) <———————————————————-
  DelayMs(50);                          //Attesa fine trasmissione dati verso il ricevi-
tore
  chr_temp=chr_5;                       //Visualizzazione carattere 'S' —> Spegnimento
  }
  //Trasmissione seriale attivata
  else                                  //L'else assicura che in caso di valori diversi da
0 e 1, il flag riassuma uno di questi due valori
  {
  transmit_en_flag=1;                   //Abilitazione flag
  chr_temp=chr_A;                       //Visualizzazione carattere 'A' —> Accensione
  }
  //Salvataggio in EEPROM di transmit_en_flag (transmit_en_flag)
  EEPROM_WRITE(EEADD_transmit_en_flag,transmit_en_flag);//Initiate writing value to address
  while(WR) continue;                   //wait for end-of-write before EEPROM_READ
  WREN=0;                               //Scrittura disabilitata
  //Visualizzazione carattere: 'A' per Accensione Display o 'S' per Spegnimento Display
  for (counter=0;counter<3;counter++)
  {
  DelayMs(250);                         //Ritardo
  DISP=chr_temp;                        //stampa carattere
  DelayMs(250);                         //Ritardo
  DISP=clear;
  }
}
//*******************************************************************************************
```

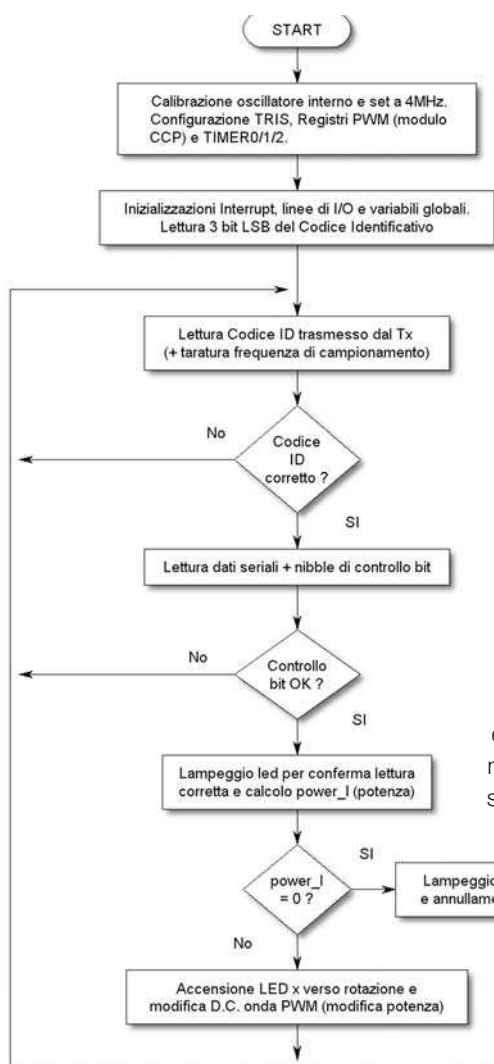



Figura 5: flowchart del firmware del Ricevitore.

START RF 1 0 0 1 0 0 1 1 0 1 1 1 1 0 1 1 0 1 1 0 1 0 1 0 0 1 1 1 END
0 1 1 1 END

Figura 4: esempio di comunicazione seriale tra Tx ed Rx.

contenente la *main()* e le funzioni di supporto quali: *read_serial_data()* per la lettura delle sequenze binarie contenenti i dati utili, la *read_codeID()* per il riconoscimento del codice identificativo, e poi tutte le funzioni di ritardo già viste per il Tx. Anche in questa occasione si fa una descrizione del codice sorgente iniziale, per poi proseguire col firmware che gira sul microcontrollore PIC16F88.
Al solito ritroviamo l'inclusione dei file di libreria, della configuration bits che abilita l'oscillatore interno e seleziona la linea RBO come uscita dell'onda PWM, la dichiarazione di tutte le variabili globali e dell'array del codice identificativo, nonché la definizione dei nomi mnemonici delle linee di I/O utilizzate. Adesso esaminiamo la **figura 5** riportante il flowchart del firmware. All'accensione il micro inizia col calibrare l'oscillatore interno e a configurarlo per una frequenza di lavoro pari a 4MHz. Questa prima operazione si realizza con i bit TUN5/4/3/2/1/0 e la configurazione del postscaler (bit IRCF2/1/0). Di seguito si configurano i registri TRISA e TRISB e si attivano le resistenze di pull-up integrate (bit di configurazione RBPU).
A seguire vengono settati i registri del modulo CCP (Capture Compare PWM) relativi alla generazione dell'onda PWM a frequenza 1KHz, ovvero PIR2, TIMER2 eccetera, e poi i TIMER0/1 usati per la generazione dei diversi ritardi. Si fà presente che questo firmware prevede una routine di interrupt, usata esclusivamente per la generazione di tempi di attesa relativamente lunghi durante la normale esecuzione del firmware stesso.
Una volta inizializzati gli interrupt utili, le linee di output e le variabili globali, vengono lette le tre linee RB7/6/5 per la configurazione dei 3 LSB del codice identificativo. A questo punto la fase di accensione ha termine ed inizia quel loop infinito che svolge tutte le funzioni del ricevitore. Il code inizia con la chiamata della routine *read_code_ID()* che rileva il codice identificativo e lo confronta con quello presente in memoria.
Questa routine inoltre, consente la taratura della frequenza di campionamento del segnale seriale emesso dal modulo ricevente RF, qualora venisse avviata la relativa procedura. In caso di riconoscimento si prosegue con l'esecuzione della read_serial_data(), con la quale vengono lette le sequenze binarie contenente l'informazione. Come già spiegato, la sequenza dei dati ha termine con un nibble per la verifica dei bit, ugualmente a quanto accade per il Tx. Se il confronto ha esito positivo il led di segnalazione emette un lampeggio a conferma.
Una volta letti e memorizzati i parametri di pilotaggio del motore (i 10 bit per il D.C. dell'onda PWM, rot_an e rot_or), si calcola il livello di potenza richiesto per l'uscita (power_l) e si configurano i registri del PWM. Assieme alla modifica della potenza in uscita vengono anche pilotati i led per la segnalazione del verso di rotazione, in base ai bit rot_an e rot_or appena letti. Nel caso in cui power_l (ovvero la potenza richiesta dall'utente) dovesse risultare nulla, immediatamente il D.C. dell'onda PWM viene annullato per annullare così la potenza erogata al motore, e vengono fatti lampeggiare continuamente e contemporaneamente Led_an e Led_or. Questo evento accade nelle seguenti due circostanze: l'utente ha richiesto il blocco di emergenza, oppure la potenza richiesta è stata annullata tramite la manopola o i tasti di incremento/decremento della scheda base.

CONCLUSIONI

Come discusso nei precedenti articoli, il codice identificativo del sistema può essere personalizzato via hardware per i soli 3 bit meno significativi.
Questo significa che in uno stesso ambiente, se non si manifestano fenomeni di interferenza tra due o più segnali radio, è possibile far funzionare contemporaneamente fino ad 8 controlli remoti. Il lettore comunque ha la possibilità di personalizzare via software tale codice e magari di spostare i tre bit su posizioni più significative.
Si fa notare inoltre, che il project del trasmettitore richiede l'inclusione del file header *pic12f6x.h*, mentre quello del ricevitore il file *pic16f87.h*. ❏

CODICE MIP 2756775

di MASSIMILIANO MIOCCHI

Progettare gli alimentatori *(parte quinta)*

STABILIZZAZIONE con il BJT

Le caratteristiche di un alimentatore stabilizzato migliorano fortemente se la stabilizzazione viene fatta con un transistore BJT anziché un diodo Zener. Ecco come dimensionare il circuito

Negli articoli precedenti è stato già anticipato come in un sistema lineare di alimentatore a forti correnti, sia impossibile o meglio sconsigliabile l'utilizzo del diodo zener, sia per la dissipazione a cui sarebbe sottoposto sia per la stabilizzazione dello stesso in prossimità della sua caratteristica voltamperometrica.
A questo limite viene proposta una soluzione di gran lunga superiore sia per affidabilità ,dissipazione e controllo della stabilizzazione con l'impiego di un semiconduttore controllato come il BJT. Il BJT viene utilizzato in due configurazioni principali, configurazione in serie al carico, o in parallelo al carico detto anche di shunt. La **figura 1** mostra le due configurazioni possibili. In entrambe le configurazioni, il BJT deve necessariamente lavorare in zona attiva, questo per consentire al dispositivo e di conseguenza al sistema di stabilizzazione di poter essere controllato in modo lineare. Per zona attiva intendiamo l'insieme dei punti della retta di carico posta fra il punto A e B rappresentati nella **figura 2**. E' noto come un BJT operi in zona attiva, nella **figura 2** sono presente n curve per ogni valore della corrente di base attribuito ad un particolare valore della corrente di collettore Ic e da un valore della tensione Vce, stabilizzare una tensione Vo di uscita in un alimentatore non è cosi scontato, non basta decidere un punto di lavoro adeguato del BJT, ma occorre fare delle considerazioni preliminari. Affinché si possa ottenere un buon rendimento nella stabilizzazione della tensione di uscita occorre neutralizzare tutte quelle cause che possono determinare un instabilità nel processo di stabilizzazione della tensione di uscita, che evitino uno scostamento

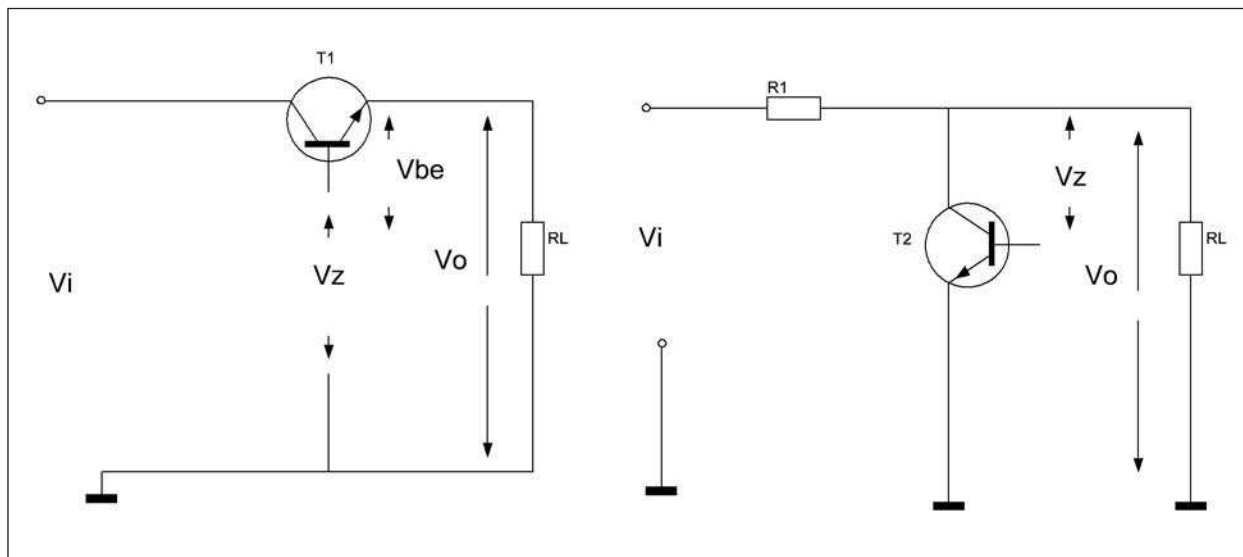



Figura 1: stabilizzazione serie e parallelo.

della tensione di uscita dal suo valore nominale. Queste cause di instabilità sono riconducibili alla possibilità di fluttuazione della tensione di ingresso Vi non ben filtrata dalla capacità di filtro, variazioni della corrente di uscita Il che viene richiamata dal carico, dai parametri dei componenti attivi usati nel sistema dovute alle escursioni termiche. Per questi motivi si rende necessario al momento del progetto, prendere in esame tutti questi fattori di possibili instabilità che possono essere riassunti dall'equazione che segue:

$$\Delta V_0 = S_v \Delta V_i + R_0 \Delta I_L + S_T \Delta T \quad \textbf{[1]}$$

Dalla [1] si possono dedurre i singoli termini

$$S_v = \left[\frac{\Delta V_0}{\Delta V_i}\right]_{I_L=K;T=K}$$

Sv è il rapporto tra la variazione della tensione di uscita dovuta ad una variazione della tensione di ingresso Vi e la correlata ΔVi per una corrente Il e un fattore di temperatura T costanti, con fattore di proporzionalità K detto anche coefficiente di stabilità di tensione costante. Analogamente determiniamo i restanti fattori dall'equazione [1] e troviamo:

$$R_0 = \left[\frac{\Delta V_0}{\Delta I_L}\right]_{V_i=K;T=K}$$

Dove Ro è la resistenza di uscita dello stadio stabilizzatore ed è espressa dal rapporto, per Vi e T costanti, tra la variazione ΔVo della tensione di uscita e la corrente assorbita dal carico sempre espressa come ΔI_L. Infine ricaviamo il coefficiente di temperatura, dato come rapporto tra la variazione della tensione di uscita ΔVo e la variazione della temperatura ΔT:

$$S_T = \left[\frac{\Delta V_0}{\Delta T}\right]_{V_i=K;I_L=K}$$

E comprensibile come per ottenere un ottimo fattore di stabilità della tensione di uscita, i coefficienti pocanzi determinati, debbano assumere valori più piccoli possibili, al fine di ottenere una variazione della tensione di uscita ai capi del carico pressoché nulla(solo idealmente).

BJT IN SERIE AL CARICO

Una delle più semplici configurazioni di BJT in serie al carico è quella raffigurata in **figura 3**, si vede come la base del transistor in configurazione a collettore comune, sia affidata alle proprietà di un diodo zener.
E' la classica semplificazione di uno stadio stabilizzatore, ma sufficiente a chiarire molto bene il concetto. La configurazione a collettore comune, o emitter follower, presenta un elevata resistenza di ingresso e una bassissima resistenza di uscita, in accordo con ciò che è stato espresso in precedenza relativamente alla resistenza di uscita di un alimentatore.
Quest'ultima è in pratica il carico Rl. In questa configurazione possiamo offrire in uscita una tensione Vo costante a prescindere dalle variazioni della tensione di ingresso ΔVi, o a fronte di una variazione del carico Rl, con conseguente variazione della corrente da esso assorbita ΔI_L. Si faccia riferimento alla maglia di uscita dello stadio stabilizzatore, ricavando la tensione di uscita Vo e la tensione di zener Vz:

$$V_0 = V_{be} + V_z \quad \textbf{[2]}$$

Dai concetti espressi nel corso sul diodo zener, sappiamo che Vz(tensione di zener) è costante, ne segue che un incremento della tensione di uscita Vo necessariamente provoca una diminuzione della tensione Vbe, ciò rende il BJT meno conduttore determinando un aumento della tensione Vce, riportando la tensione di uscita Vo al valore iniziale.
Analogamente se la tensione di uscita Vo diminuisce o subisce un decremento dovuto ad un eccessivo aumento della corrente assorbita, in questa situazione ipotetica avremo che la tensione di giunzione Vbe tenderà ad aumentare portando a maggiore conduzione il BJT con una conseguente diminuzione della tensione Vce, causando cosi un aumento

della tensione Vo riportandola ai valori iniziali. Tutto ciò sempre considerando la tensione di zener costante. Osserviamo come il BJT stia lavorando come una resistenza variabile in serie al carico, regolandosi in modo opportuno a seconda che la tensione di uscita Vo, aumenti o diminuisca. In questa analisi abbiamo considerato Vo, la tensione di uscita, come variabile del sistema, ma possiamo considerare e dobbiamo farlo, che anche la tensione di ingresso subisce variazioni più o meno consistenti.
Supponiamo che la tensione di ingresso Vi subisca delle variazioni, abbiamo sempre la tensione di zener Vz costante, quindi ad un incremento della Vi, porterebbe un aumento della corrente che circola nella resistenza di caduta Rb del diodo zener e nel diodo stesso, senza andare a modificare in modo considerevole il valore di K, cioè del coefficiente di stabilità.
Resterà costante il valore della tensione Vbe e quindi del punto di lavoro del BJT e di conseguenza del valore della tensione di uscita Vo, idem se facessimo l'analogia con un decremento della tensione di ingresso.
Chiaramente quando mi riferisco ad un incremento o decremento della tensione di ingresso, è ben inteso in un margine di tolleranza molto stretto.

LE FORMULE DI PROGETTO IN SINTESI

In ambito pratico spesso si fa uso di formule pratiche che ci consentono di risolvere problemi legati a molte situazioni progettuali di diversa difficoltà. Le formule utili scaturiscono da ciò che abbiamo fin qui esposto e a volte ci consentono di far uso di fattori numerici che vengono direttamente dall'esperienza pratica. Possiamo suddividere in diversi step le fasi di dimensionamento di uno stadio stabilizzatore:
1- Tensione Vi di ingresso: per una sufficiente stabilizzazione della tensione di uscita, la tensione di ingresso Vi deve corrispondere ad un valore pari a 1,2 – 2 volte il valore della tensione di uscita Vo, in formula:

$$V_i = (1,2 \pm 2)V_0 \quad [3]$$

2- Scelta del transistor BJT, il quale deve fornire una corrente assorbita dal carico Il pressoché coincidente con la corrente di collettore Ic, dissipando una potenza espressa dalla seguente relazione:

$$P_Q = (V_i - V_0) I_{L\,max} = (V_i - V_0) I_{c\,max} \quad [4]$$

3- Un aspetto fondamentale è la scelta del diodo zener, quindi la tensione di zener Vz, che determina il valore della tensione di riferimento.
Di conseguenza possiamo utilizzare per il nostro scopo la relazione [2] che ricordiamo essere Vo=Vbe+Vz attribuendo alla Vbe un valore pari a Vbe=0.65V per BJT al silicio, contrariamente per BJT al germanio la tensione Vbe=0,3V. analogamente si determina la potenza che esso deve dissipare o meglio che sia in grado di dissipare, esprimendola come dalla seguente relazione:

$$P_z = V_z I_{z\,max} \quad [5]$$

Dove la Iz_{MAX} è espressa come rapporto tra la differenza della tensione di ingresso Vi e la tensione di uscita Vo, e il valore della resistenza di polarizzazione Rb, in formula:

$$I_{z\,max} = \frac{V_i - V_0}{R_b} \quad [6]$$

Nella [6] abbiamo introdotto la resistenza di polarizzazione Rb, la medesima può essere espressa come rapporto di una tensione e una corrente, più precisamente

$$R_b = \frac{V_i - V_0}{I_z + I_{b\,max}} \quad [7]$$

Dove Ib_{MAX} è la corrente di polarizzazione di base del BJT corrispondente ad una determinata corrente di collettore Ic, sapendo che:

$$I_{b\,max} = \frac{I_c}{hfe_{\,min}} = \frac{I_L}{hfe_{\,min}}$$

Per meglio fissare le idee su tali concetti e sulle formule in sintesi a pena esposte, consideriamo un esempio molto pratico. Prendiamo in esame il regolatore rappresentato in **figura 3**, e si consideri di voler avere una tensione di uscita di 12 volt ai capi di un carico che assorbe 300 mA, che potrebbe essere benissimo un diodo led power o una lampadina ad incandescenza a bassa tensione. Cominciamo con il calcolare il valore della tensione di ingresso, che abbiamo detto ,deve essere di 1,2-2 volte il valore della tensione di uscita da stabilizzare, quindi nel nostro caso poniamo la tensione di ingresso pari a 2volte la tensione di uscita Vo, in formula abbiamo:

$$V_i = (1,2 \pm 2)V_0 = 2V_0 = 24Volt$$

Ora si determini la potenza che deve dissipare il BJT in tali condizioni, cioè con una corrente di collettore $Ic=I_L$=300mA di conseguenza dalla formula [4] avremo:

$$P_Q = (V_i - V_0) I_{c\,max} = (24-12)0,3 = 3,6W$$

Da questo dato emerge che il BJT deve essere in grado di dissipare almeno 5W per far fronte al problema, quindi al nostro scopo può essere più che sufficiente utilizzare un classico 2N3055, capace di sopportare correnti di collettore di 700mA. Il guadagno Hfe minimo per il BJT è compreso tra 25- 250, ma relativamente alla corrente richiesta dall'esempio abbiamo un Hfe minimo di 100, da cui calcoliamo la corrente di base Ib sostituendo i valori e sapendo che la corrente di base Ib è legata alla corrente di collettore Ic e al guadagno dalla relazione:

$$I_b = \frac{I_c}{hfe_{\,min}} = \frac{0,3}{100} = 3mA$$

Quindi il diodo zener andrà scelto per una tensione data dalla relazione [2], che ricordiamo essere:

$$V_0 = V_{be} + V_z \Rightarrow V_z = V_0 + V_{be} = 12,65Volt$$

Per concludere calcoliamo la resistenza di base Rb facendo uso della formula [7], ma prima di procedere fissiamo una corrente di zener Iz pari a 15mA conoscendo la corrente di base possiamo facilmente

per sostituzione alla [7] ricavare il valore di Rb, più precisamente:

$$R_b = \frac{V_i - V_0}{I_z + I_{b\,\max}} = \frac{24-12}{0.015+0.03} = 270\Omega$$

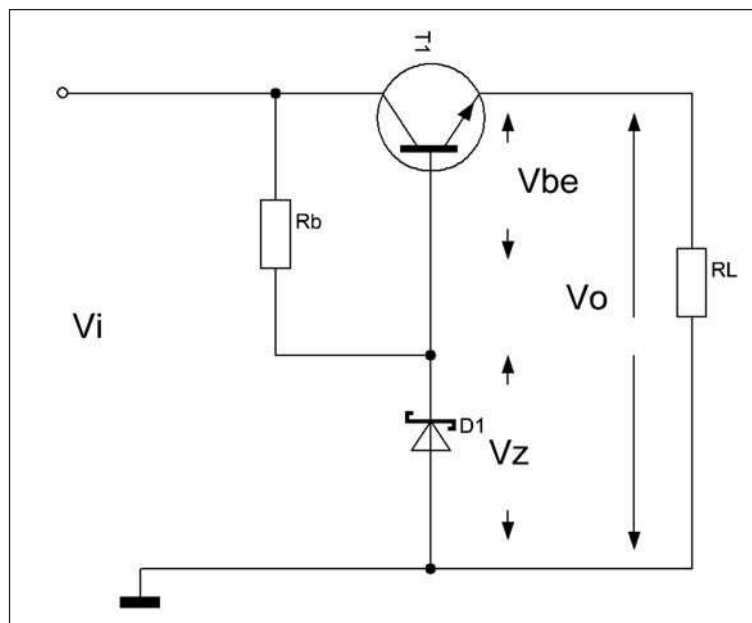
Analogamente a sopra , procediamo con il calcolare la potenza che deve dissipare il diodo zener, facendo uso della [5]:

$$P_z = V_z I_{z\,\max} = 12.65x0.015 = 190mW$$

Quindi possiamo con tranquillità utilizzare un diodo da 1W.
Un interessante applicazione di ciò che abbiamo pocanzi affrontato, è sicuramente dimensionare il sistema considerando eventuali variazioni della tensione di ingresso ΔVi, questo aspetto è interessante ed è molto frequente nella pratica ma proveremo che in un sistema lineare

come quello di **figura 3** la variazione della tensione ΔVi di ingresso non influirà in modo pesante sul valore della stabilizzazione della tensione di uscita ΔVo. Cominciamo con il dimensionare uno stadio come quello di **figura 3**.
Supponiamo di volere una tensione di uscita Vo pari a 8V, e per ipotesi la tensione di ingresso subisca una variazione ΔVi di 2V rispetto la suo valore nominale di 24V, supponendo una variazione a ribasso, avremo che la ΔVi varia tra 22V e 26V rispetto al suo valore. Supposto che anche la corrente vari con la stessa proporzione ,avremo una corrente assorbita dal carico che varia in un intervallo tra 270 - 330 mA.
Rispetto all'esempio precedente nulla è diverso se non nelle considerazioni, pro-

Figura 2: la zona attiva del transistore BJT.

Figura 3: la più semplice configurazione per uno stabilizzatore serie.

cediamo quindi con la scelta del BJT, tenendo conto però che la potenza da esso dissipata sarà in relazione al valore massimo della corrente assorbita dal carico e al valore massimo della tensione di ingresso, in formule diciamo che:

$$V_i = V_{i\,max} = 24Volt$$

$$I_L \equiv I_{L\,max} = 330mA$$

Di conseguenza la potenza P_{Qbjt} del BJT utilizzato si ricava dalla relazione [4] che tradotta sostituendo i valori numerici:

$$P_Q = (V_i - V_0)\, I_{L\,max} = (V_i - V_0)\, I_{c\,max} = 5.8W$$

Siamo ben al di sotto della potenza dissipabile dal BJT in esame. Si passa alla scelta del diodo zener che in relazione alla tensione di zener per ipotesi imposta a 8 volt, risulta in base alla relazione Vo=Vbe+Vz=6,65V. La scelta è chiaramente mirata ad un valore più prossimo della tensione di zener che è di 8,7Volt con una potenza dissipabile di 1W, e fissando per convenzione una corrente di zener Iz pari a 10mA.
Sapendo che la $I_L=I_{Lmax}$=330mA e avendo posto un hfe_{MIN}=100 la corrente di polarizzazione del BJT, la Ib, sarà di:

$$I_b = \frac{I_{c\,max}}{hfe_{\,min}} = \frac{I_{L\,max}}{hfe_{\,min}} = 3mA$$

Avendo per ipotesi supposto una variazione della tensione di ingresso Vi di circa il 20%, occorre mantenere il valore della corrente di zener Iz al valore di 10mA anche quando il valore della tensione di ingresso scende al suo valore minimo che è di 22Volt. Quindi dalla [7] ricaviamo il valore della resistenza Rb di polarizzazione secondo la relazione:

$$R_b = \frac{V_{i\,min} - V_0}{I_z + I_{b\,max}} = 1023\Omega \approx 1K\Omega$$

Effettuiamo ora un ulteriore verifica per la scelta del diodo zener, in rapporto ai valori fin qui trovati, sostituendo alla [6] la tensione di ingresso con il suo valore massimo, ritroveremo una corrente di zener Iz che sarà appunto la massima corrente di zener circolante nel diodo, tradotto in formula:

$$I_{z\,max} = \frac{V_{i\,max} - V_z}{R_b} = 14.9mA$$

E ora per finire ricalcoliamo il valore della potenza che deve essere in grado di dissipare il diodo zener, facendo ricorso alla formula [5]:

$$P_z = V_z I_{z\,max} = 130mW$$

Quindi possiamo confermare l'utilizzo di un diodo zener che sia in grado di dissipare 1W.

CONCLUSIONI

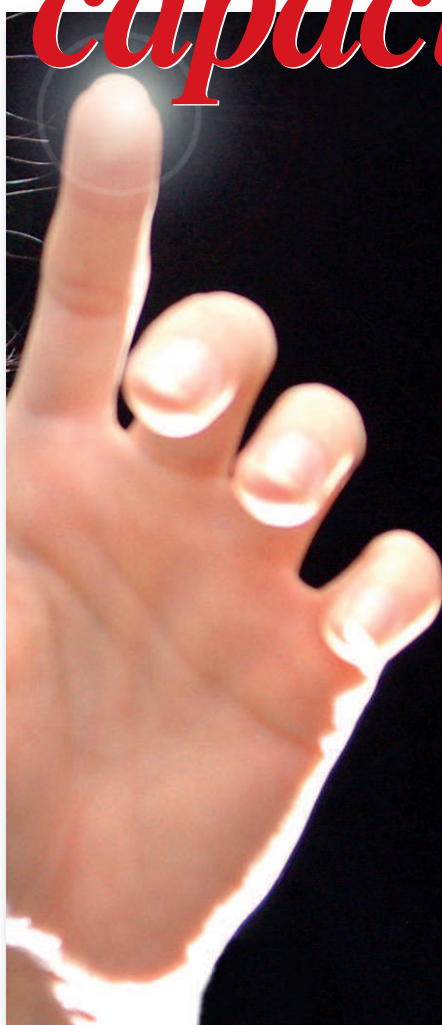
Abbiamo visto come sia possibile realizzare uno stadio di stabilizzazione con l'ausilio di un BJT,la cui funzione è quella di resistenza variabile,e del diodo zener il cui compito è quello di tenere costante nell'intorno del suo valore la tensione di zener Vz.
Nei concetti espressi relativi alla stabilizzazione, si è visto come la peculiarità del diodo zener è quella di mantenere costante il valore nominale della tensione di zener al variare della sua corrente, o più precisamente per piccole variazioni della medesima.
Si comprenda come sia inaffidabile lo stadio di stabilizzazione pocanzi esaminato se decidessimo di dimensionarlo per correnti superiori, e per superiori intendo oltre l'ampere, questo porterebbe ad un abbassamento eccessivo della Rb, quindi ad un aumento della corrente Iz di zener con la conseguente instabilità del diodo zener nel tenere costante il valore della tensione di zener Vz, oltrepassando il gomito visibile nella sua caratteristica volt-amperometrica. ❑

CODICE MIP 2756759

di MARCEL FLIPS

Utilizzare il PICkit 2 Debug Express

INTERRUTTORE *capacitivo*

Il PICkit™ 2 Debug Express di Microchip è un piccolo kit dotato di debugger e scheda di sviluppo. Esso dispone di tutto il necessario per realizzare un piccolo progetto partendo da zero. Il progetto presentato in queste pagine è relativo a uno switch capacitivo costruito ricorrendo solo a un paio di componenti passivi. Lo switch è implementato con la tecnologia mTouch™ di Microchip

Microchip attualmente offre numerose soluzioni hardware adatte alle esigenze di qualsiasi applicazione, dal piccolo PIC10F per schemi base a tasto singolo fino alle famiglie di MCU Mid-Range e PIC24FJ, estremamente ricche di periferiche. Il progetto qui presentato è realizzato con il modello PIC16F887, che equipaggia la scheda di sviluppo.

mTOUCH

Le soluzioni mTouch™ di Microchip sono prodotti di rilevamento tattile che sfruttano la variazione capacitiva. Molte applicazioni moderne ricorrono al rilevamento capacitivo per assicurare un look elegante, piacevole e professionale. La **figura 1** illustra le caratteristiche base del rilevamento capacitivo. Quando un utente pone il dito in prossimità dell'area di rilevamento introduce nel sistema di una certa capacità aggiuntiva. Questo incremento di capacità viene rilevato attraverso un microcontroller PIC®, supportato da una circuiteria aggiuntiva e da un opportuno software che individua l'effettiva pressione del tasto. Il PIC16F887 è equipaggiato con un comparatore dotato di Latch SR. La soluzione capacitiva mTouch™ provvede la capacità del sensore, utilizzata come elemento per determinare la frequenza di un oscillatore a rilassamento. Il software si incarica di rilevare e validare qualsiasi variazione di frequenza dell'oscillatore indotta da un intervento dell'utente. La **figura 1** mette in evidenza il circuito oscillatore/timer e la capacità del sensore. L'oscillatore a rilassamento è un oscillatore RC che sfrutta 2 comparatori e un latch SR per variare, verso l'alto o verso il basso, la direzione di carica della tensione del condensatore di rilevamento. Esso caricherà e scaricherà il condensatore a un tasso determinato dalla costante di tempo RC, portando la tensione ai valori superiore e inferiore fissati dagli ingressi positivi dei comparatori. Il limite superiore è interno, ma il limite inferiore deve essere fornito esternamente. A tale scopo sarà sufficiente un semplice divisore a resistenza associato a un condensatore di disaccoppiamento da 100nF. Il condensatore da 100 nF è permette di eliminare il rumore ad alta frequenza proveniente dall'alimentazione, assicurando un limite inferiore più stabile. La tensione sul condensatore di rilevamento varierà all'in-

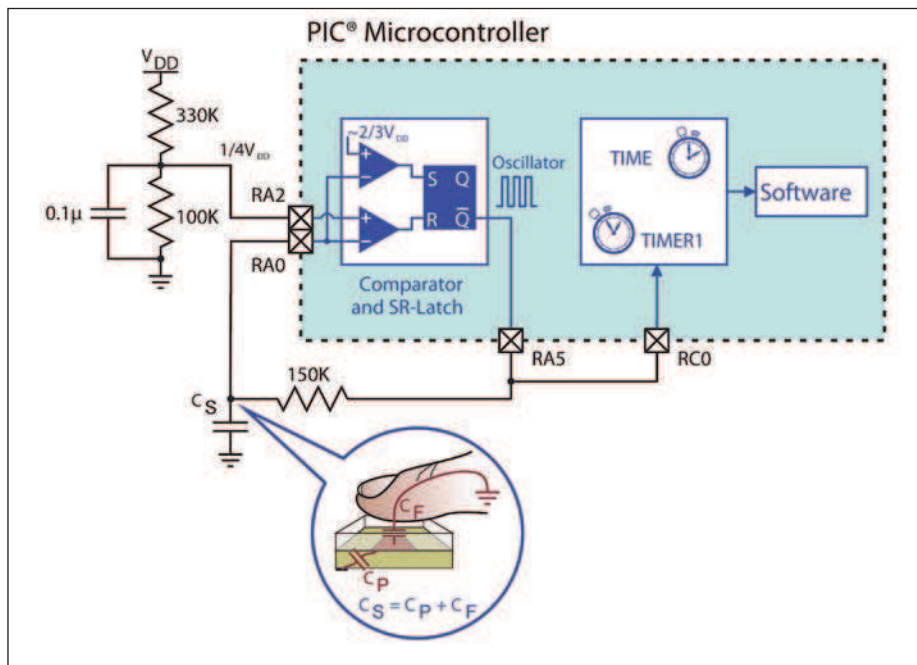



Figura 1: schema base dell'oscillatore mTouch.

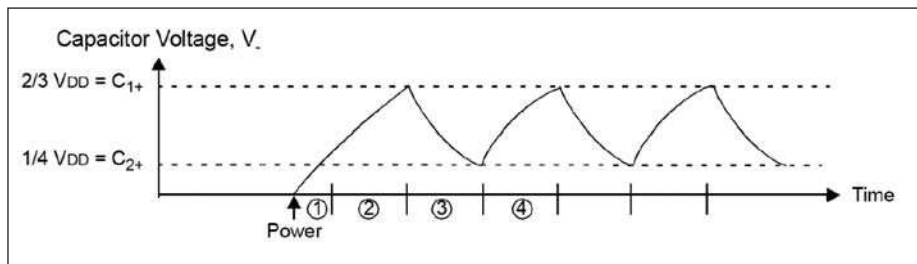



Figura 2: cicli di carica e scarica.

Figura 3: componenti per la determinazione del limite di tensione inferiore.

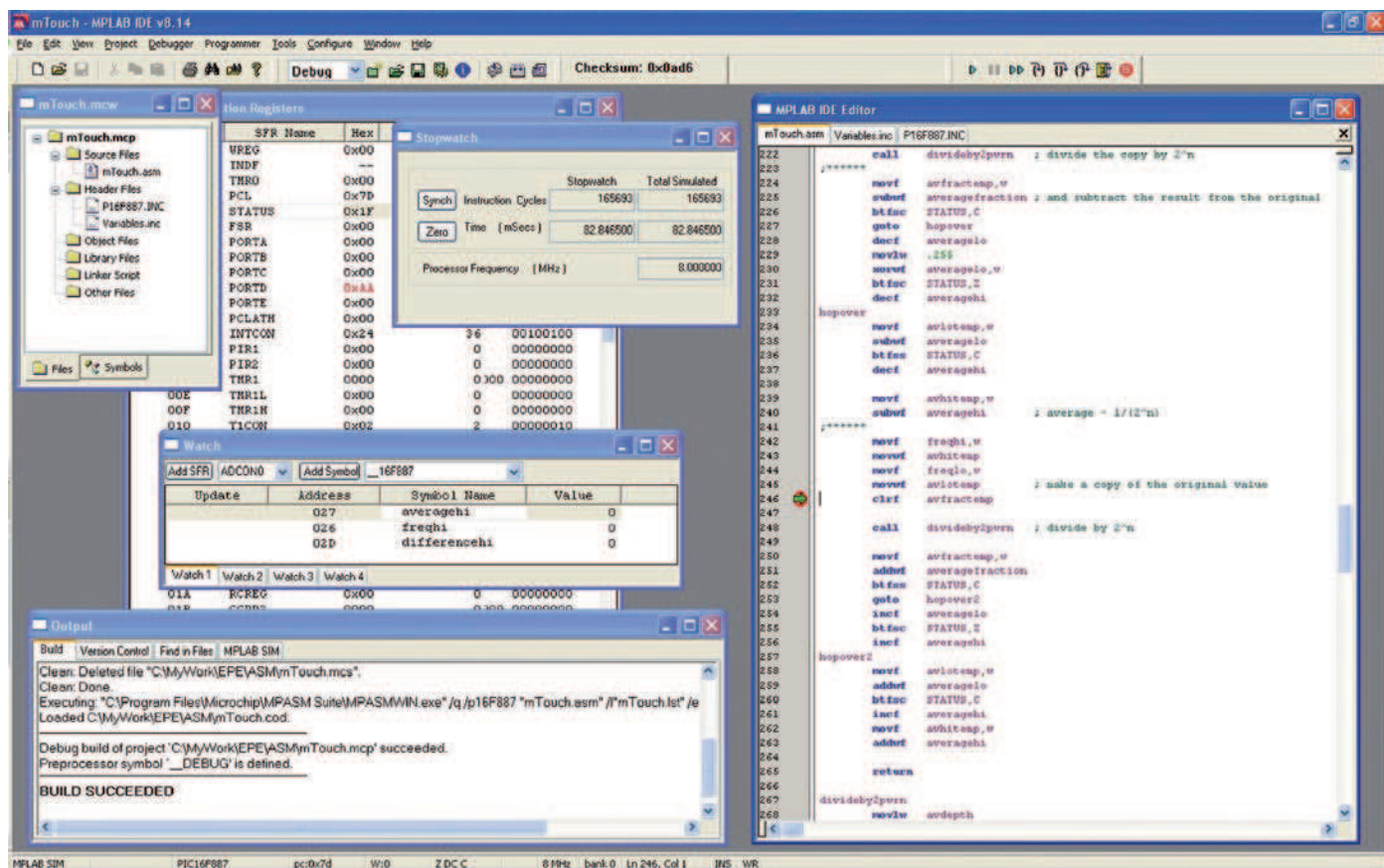
Figura 4: superficie di rilevamento mTouch offerta da una moneta da 20 cents.

terno dei limiti fissati: essa è pilotata dai segnali di livello logico a C2OUT. Al fine di ottenere il comportamento di carica e scarica desiderato, l'uscita del comparatore 2, C2OUT, è configurata come Q. Il resistore di feedback insieme all'area del sensore - denotata da Cs - costituiscono il circuito RC. Quando la tensione sul condensatore Cs è sotto il limite inferiore, C2OUT va alto e il sistema inizia a caricarsi. Tra i due limiti, il sistema mantiene l'ultimo stato (caricandosi o scaricandosi). Quando la tensione è sotto il limite superiore, C2OUT va basso e il sistema inizia a scaricarsi. Un'illustrazione del ciclo di carica e scarica è riportata in **figura 2**. L'uscita Q e lo stato di carica o scarica sono determinati dai valori relativi dell'ingresso negativo all'ingresso positivo di ciascun comparatore e dal latch SR.

COSTRUZIONE DEL CIRCUITO

Il circuito può essere realizzato nel giro di pochi minuti. Per sviluppare la tensione per il limite inferiore l'autore ha utilizzato un condensatore e due resistenze a montaggio superficiale. I componenti sono stati saldati sull'area di prototipazione **(figura 3)**. Per connettere la tensione al pin RA2 del microcontroller PIC è stato utilizzato un filo da cablaggio. Quando si progetta un tasto capacitivo, la forma dell'area non è molto importante. L'area è funzione del progetto. Un'area più ampia assicura però una migliore sensibilità e una migliore capacità di rilevamento. L'autore ha optato per l'area offerta da una moneta da 20 centesimi, ma qualsiasi superficie metallica potrebbe andare bene. In **figura 4** è possibile notare la resistenza da 150K utilizzata per la carica e la scarica. Il valore della resistenza e il condensatore di rilevamento determinano la frequenza dell'oscillatore. Qualsiasi frequenza nel range da 100kHz a 400kHz andrà bene. La frequenza esatta non è importante ma lavorare con frequenze alte permette di sfruttare più conteggi nel processo di misura, ottenendo così una migliore risoluzione rispetto a una bassa frequenza. L'ultimo step riguarda la connessione dell'uscita dell'oscillatore al Timer1, così che il software possa misurare la frequenza. Per connettere RA5/C2OUT al pin RC0/T1CKI del microcontroller PIC è stato utilizzato un altro di filo da cablaggio. L'ultimo filo da saldare va dalla giunzione dell'RC al modulo comparatore (RA0/C12IN0-). Rimuovere R3 dalla scheda per disabilitare il potenziometro onboard **(figura 4)**.

SOFTWARE

Ora che è il circuito stato costruito, bisogna implementare il codice. Il codice è stato scritto in linguaggio assembly e può essere compilato utilizzando MPLAB®, scaricabile gratuitamente dal sito web Microchip. Il primo passo è l'inizializzazione della periferica Comparatore. I registri sono predisposti dalla subroutine Init. Una volta che l'oscillatore è in funzione, la

sua frequenza deve essere controllata per rilevare l'eventuale caduta dovuta alla presenza del dito. Ogni volta che C2OUT cambia da '0' a '1', Timer1 s'incrementa. Per misurare la frequenza in un periodo definito viene utilizzata una base dei tempi fissa. Timer0 offre la base dei tempi per questo periodo fisso. All'inizio della misura, Timer0 viene azzerato: esso conterà fino a 255, quindi andrà in overflow. Sull'overflow, Timer0 genera un interrupt, T0IF, che porterà il programma alla Interrupt Service Routine indicata dal vettore. A questo punto viene letto il valore di TMR1 e viene confrontato con le letture precedenti. Se il valore corrente di TMR1 è significativamente più basso vuole dire che la capacità è aumentata, indicando la pressione del tasto. I LED connessi a PORTE si illumineranno. Il nuovo valore viene inserito in una media utilizzata come valore base per il confronto. Alla fine della Interrupt Service Routine, quando tutte le task che determinano la pressione del tasto sono terminate, sia Timer1 sia Timer0 vengono azzerati e riavviati per la lettura successiva. Per iniziare a progettare è possibile fare riferimento all'offerta speciale extra di Microchip riportata nella pagina successiva!

INFORMAZIONI AGGIUNTIVE

Ulteriori informazioni su mTouch sono disponibili nel sito web Microchip: www.microchip.com/mtouch.
Qui sono scaricabili schemi circuitali, note applicative, manuali, risorse di eLearning, tool di sviluppo, librerie di codice gratuito, software diagnostici e raccomandazioni sui prodotti. Se desiderate aumentare le vostre capacità di progettazione, è possibile ricorrere a uno dei tre sistemi di sviluppo mTouch™
Appartenente alla PICDEM™ Touch Sense Demonstration Board, il kit gratuito di sviluppo software mTouch prevede il tool di Windows-based mTouch Diagnostic, uno strumento dotato di interfaccia grafica che permette di analizzare le informazioni in tempo reale.

PICDEM Touch Sense 1 Development Kit – DM164125
Questo kit permette di sperimentare la tecnologia di rilevamento tattile utilizzando tasti e cursori. Il tutto prevede l'impiego del microcontroller a 8-bit PIC16F con modulo latch S/R: il PICkit Serial Analyzer è incluso.

PICDEM Touch Sense 2 Development Kit – DM164128
Questo kit permette di sperimentare la tecnologia di rilevamento tattile utilizzando i prodotti della famiglia a 16-bit PIC24F, dotati di Charge Time Measurements Unit (CTMU).

PIC24F Starter Kit - DM240011
Questo kit a basso costo include una scheda, l'ambiente di sviluppo integrato MPLAB, il compilatore C MPLAB C30 e tutto il necessario per iniziare a progettare un'applicazione di rilevamento tattile. Questo economico kit prevede anche un debugger–programmer in-circuit integrato, connettori per host e dispositivi USB, Led a tre colori, un touch pad capacitivo e un display OLED. Il software di sperimentazione - guidato da menu - supporta applicazioni di data logging, thumb drive e grafiche che permettono di testare l'MCU PIC24F. ❑

CODICE MIP 2765258

UN'OFFERTA MOLTO SPECIALE PER I LETTORI DI FARE ELETTRONICA!

Possibilità di acquistare un Debug Express Kit PICkit™ 2 Microchip

Sconto speciale 40%

Fare Elettronica e Microchip hanno stretto un'intesa per offrire ai lettori la possibilità di acquistare un Debug Express Kit PICkit™ 2 (DV164121) a un prezzo notevolmente scontato. In questo numero di Fare Elettronica avete la possibilità di aquistare il kit attraverso MicrochipDIRECT ad un imperdibile 40% di sconto! Trattandosi di un prezzo senza precedenti, l'offerta è limitata ad un prodotto per recapito (soli indirizzi italiani) / per indirizzo email.

Il PICkit 2 è un completo programmer & debugger di dimensioni tascabili che consente di facilitare lo sviluppo in-circuit sui microcontroller PIC®. Utilizzando lo strumento di sviluppo MPLAB® IDE fornito a corredo sarà possibile trasformare qualsiasi personal computer in una potente stazione di progettazione per le attività di scrittura del codice, di debugging e di programmazione dei dispositivi. Per interfacciarsi con il PICkit 2 e controllare il vostro progetto PIC è necessaria una sola porta USB: l'ambiente vi permetterà di bloccare, fare avanzare in single step e gestire i break point del codice della vostra scheda target.

Il Kit include la scheda demo per microcontroller Flash PIC16F887 a 44-pin di Microchip, il programmatore PICkit 2, il cavo USB e i CD software con l'ambiente di sviluppo integrato Microchip MPLAB IDE, con la demo del compilatore C CCS PCM™ Midrange per PIC16F887 e con il compilatore C HI-TECH PICC™ LITE. Il tutto vi metterà in condizioni di iniziare a sviluppare con la massima facilità un progetto di controllo embedded.

Come acquistare il vostro Debug Express Kit PICkit 2 a un prezzo speciale:

- Cliccare su www.microchipdirect.com, registrarsi o eseguire il log on al sito con la bandierina italiana ed inserire il part number DV164121.
- Aggiungere il PICkit al carrello e inserire il numero di Coupon FEPIC209: premere il tasto di validazione del coupon.
- Scegliere la casella che vi porterà ai processi di gestione dell'indirizzo di spedizione e del pagamento: ricordarsi di accettare i termini e le condizioni quindi confermare l'ordine.
- Autorizzare i 28 giorni per la consegna.

Condizioni: questa offerta è valida per i soli lettori italiani. È permesso un solo Debug Express Kit PICkit™ 2 per recapito o per indirizzo e-mail. Questa offerta è valida per tutti gli acquisti eseguiti prima del 31 luglio 2009 via microchipDIRECT e corredati dal corretto riferimento del coupon. L'offerta non può essere utilizzata in congiunzione ad altre offerte. Il coupon non ha valore monetario. L'offerta può essere cancellata senza preavviso. Il coupon deve essere utilizzato in ossequio a tutti i termini e le condizioni specificati da Microchip Technology Inc.

di NICOLA DE CRESCENZO

CHIAVE ELETTRONICA

Vi presentiamo una semplice chiave elettronica, con la quale potrete azionare svariati sistemi, come ad esempio, una elettro-serratura o un impianto di allarme. Il firmware di controllo vi consentirà di poter modificare con facilità la combinazione durante il funzionamento del circuito, senza alcuna modifica hardware

Lo schema proposto, utilizza una minima configurazione hardware costituita da un microcontrollore PIC16F84 e da un flip-flop. Il controllore gestisce una tastiera a matrice, occupandosi della scansione dei tasti attivati, della memorizzazione e del riconoscimento della combinazione numerica. L'integrato 74LS74, che al suo interno contiene due flip-flop di tipo D, funziona in modalità toggle e ad ogni impulso prodotto del PIC, in caso di combinazione inserita, inverte il suo stato. All'uscita del flip-flop è interfacciato, tramite un transistor BJT, un relè elettromeccanico che consente al circuito di pilotare dei carichi di potenza.

TASTIERA A MATRICE

Il Key logger utilizza una tastiera a matrice 3 x 4 del tipo riportato in **figura 2**. I collegamenti a matrice dei pulsanti sono riportati sulla parte posteriore della tastiera tramite dei pin con passo 2,54 mm. Lo standard della connessione a matrice è molto utilizzato in quanto consente di ridurre il numero dei cavi necessari per il cablaggio della tastiera: se normalmente per collegare 12 tasti sarebbero necessarie 13 linee (una per ogni tasto più una comune), sfruttando una tastiera a matrice si utilizzano solo 7 linee: 3 per le colonne + 4 per le righe **(figura 3)**. Il principio di funzionamento è molto semplice: si dispongono i tasti secondo p righe e q colonne in modo che, alla pressione del generico tasto p,q, si abbia il collegamento della riga p con la colonna q. Per gestire una tastiera di questo tipo, il principio è il seguente: si invia ciclicamente un particolare livello logico (alto o basso) ad una sola delle colonne e si controllano le righe. Quando una riga presenta quel tale livello si deduce che è stato premuto uno dei tasti di quella particolare riga; sapendo qual'è l'unica colonna attiva in quel momento, si sa quale è il tasto premuto. In **figura 4** è riportata la piedinatura del PIC16F84. Il controllore viene utilizzato alimentandolo ad una tensione continua di 5 Volt. L'oscillatore, di tipo XT è realizzato con un quarzo da 4 Mhz e 2 condensatori ceramici da 22 pF. La tastiera a matrice è collegata sui pin RA0÷RA3 del PORTA e i pin RB4÷RB6 del PORTB. L'uscita RB1 attiva un LED di segnalazione che indica la modalità "cambio codice" con scrittura in EEPROM della nuova combinazione.

L'uscita RB2 fornisce un segnale di clock che attiva il FF di tipo D nel momento in cui viene inserita la chiave numerica di attivazione. L'uscita RB3 attiva invece un buzzer in corrispondenza della pressione di ogni tasto numerico. In **figura 5** è riportata la piedinatura del flip-flpo di tipo D utilizzato. Il relè scelto è del tipo miniaturizzato per circuiti stampati. In **figura 6** è riportata la sua piedinatura Come si può osservare, il dispositivo può azionare, a seconda della necessità un contatto NA o NC. Facendo riferimento allo schema di **figura 1**, la resistenza R4 regola la corrente di base, in modo da garantire il funzionamento del transistor BC107 in modalità on-off, facendolo funzionare tra la zona di interdizione e la zona di saturazione. Il diodo 1N4007 posto in parallelo alla bobina del relè è invece un diodo di ricircolo, la cui funzione è quella di proteggere il transistor dalla sovratensioni che si possono avere sulla bobina durante le commutazioni di apertura e chiusura del circuito di eccitazione. In **figura 7** sono invece riportate le piedinature del transistor e del diodo LED. Per quanto riguarda il circuito della chiave elettronica completa, in **figura 8** è riportato il prototipo assemblato su bread –board durante la fase di test. In **figura 9** è riportato il prototipo assemblato su circuito stampato.

UTILIZZO DEL CIRCUITO

Attraverso la tastiera a matrice è possibile inserire una combinazione numerica di quattro cifre. La sequenza XXXX seguita dal simbolo # consente di attivare il key logger, nel momento in cui si inserisce la corretta combinazione numerica. La combinazione preimpostata alla prima attivazione del circuito è 1234#, pertanto bisognerà premere in successione in tasti 1,2,3,4 e il tasto # (cancelletto). In corrispondenza della pressione di ogni tasto numerico, il buzzer emetterà un se-

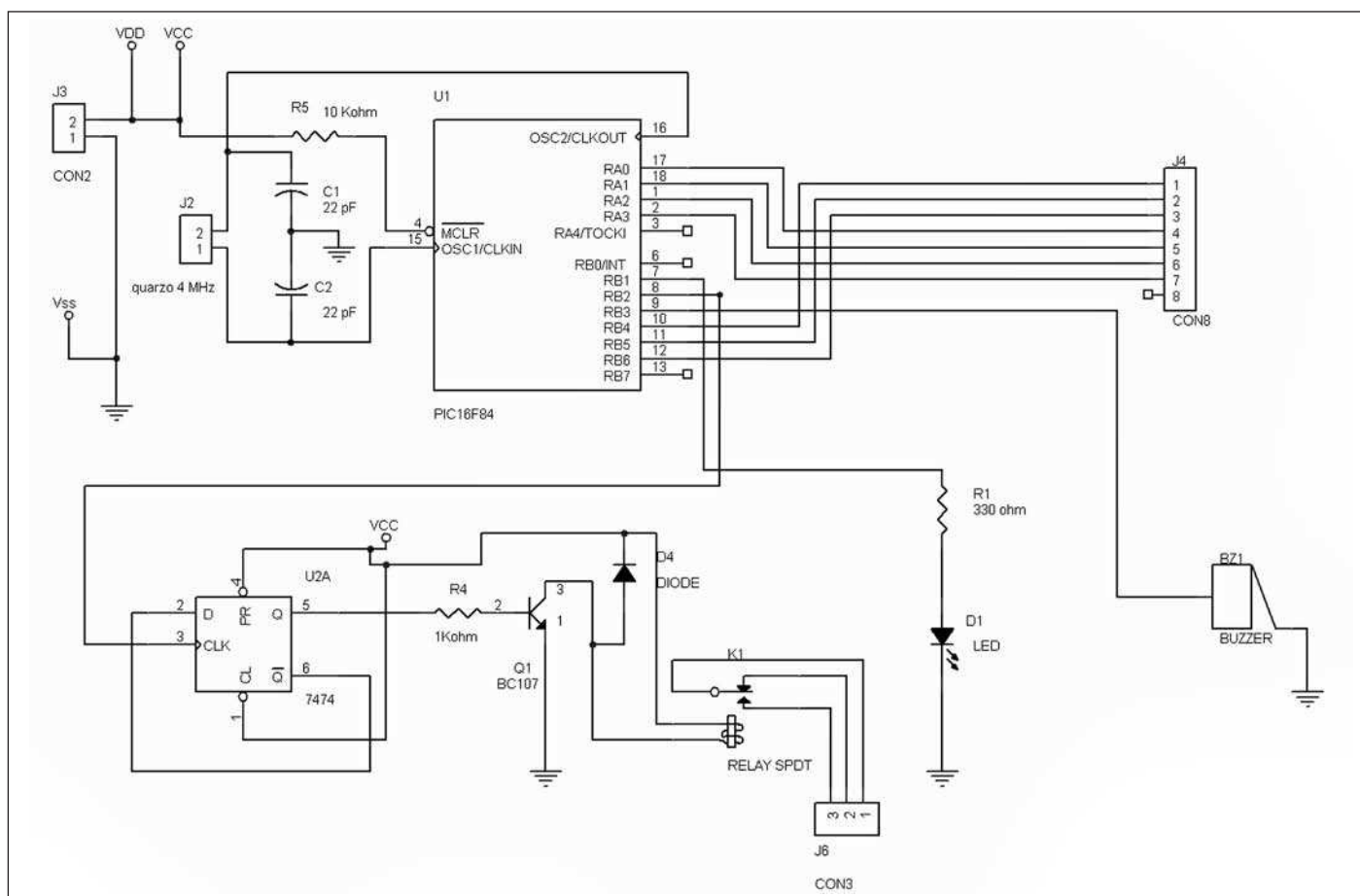



Figura 1: schema elettrico della chiave elettronica.

gnale acustico di conferma. L'immissione della combinazione determina l'attivazione del relè interfacciato con il PIC con cui sarà possibile alimentare il dispositivo da controllare.
Per disattivare la chiave sarà sufficiente immettere nuovamente la combinazione XXXX#: in questo modo il relè interfacciato al PIC sarà riportato nello stato di riposo.
La chiave elettronica permette inoltre di aggiornare la combinazione on board, ovvero durante il funzionamento del circuito: basta inserire la sequenza XXXX * (dove XXXX è la vecchia password) e in questo caso, l'accensione del LED giallo segnalerà la modalità di scrittura in EEPROM della nuova combinazione, che sarà poi salvata scrivendo YYYY#.

FIRMWARE DI CONTROLLO

La **figura 10** rappresenta il diagramma di flusso semplificato del firmware di controllo. La prima parte riguarda la confi-

LISTA COMPONENTI

R5	10 KΩ ± 5% ¼ W	U1	PIC16F84A
R1	330Ω ± 5% ¼ W	U2	74LS74
R4	1kΩ ± 5% ¼ W	Q1	BC107
C1	22 pF ceramico	D4	1N4007
C2	22 pF ceramico	J1	Quarzo 4 Mhz
C3	100 nF Poliestere	K1	Mini Relè 5,6 VCC

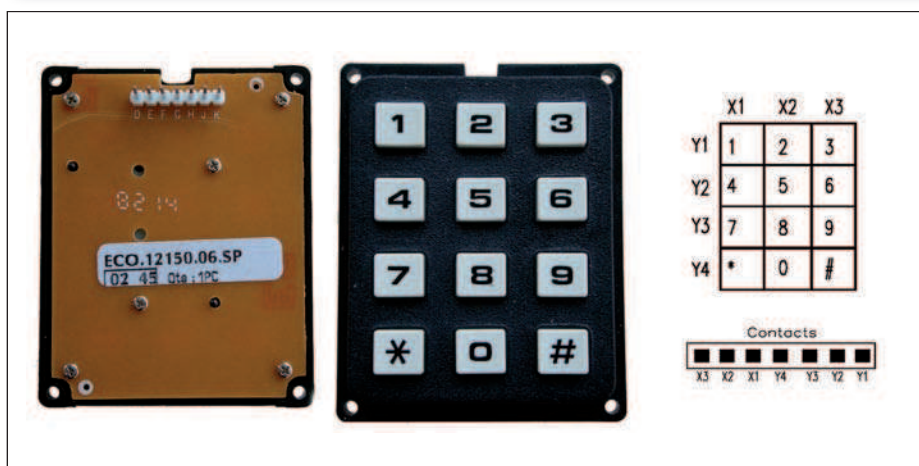



Figura 2: la tastiera a matrice con la relativa piedinatura.

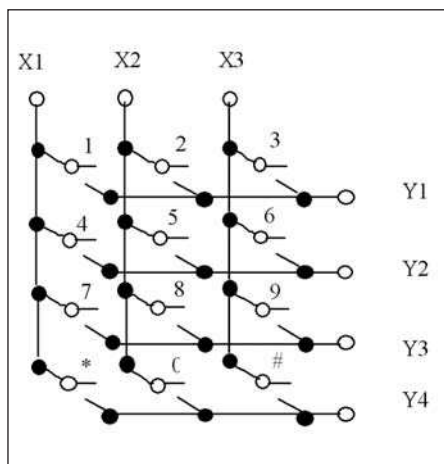



Figura 3: il principio di funzionamento della tastiera.

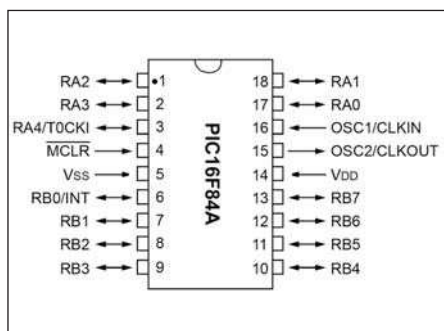



Figura 4: pinout del PIC16F84.

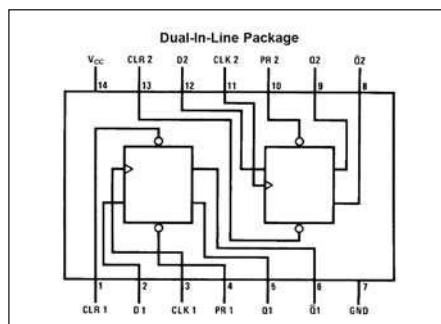



Figura 5: la piedinatura del flip-flpo di tipo D utilizzato.

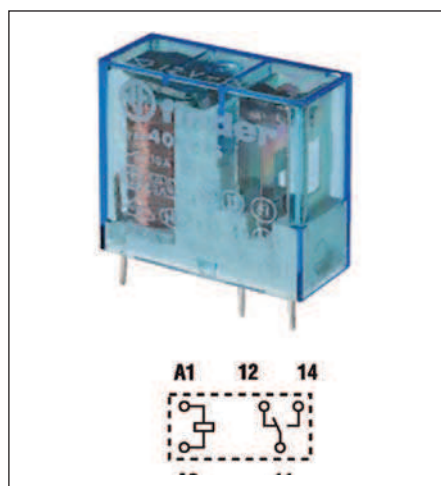



Figura 6: il relè con la relativa piedinatura.

Figura 8: il prototipo realizzato su breadboard.

Figura 7: il buzzer, il LED ed il transistor.

gurazione del dispositivo: in questa sezione, oltre alla definizione dei parametri fondamentali per il funzionamento del controllore (come ad esempio il tipo di oscillatore, il settaggio dei fuses, la mappatura della ram etc), i pin RA0- RA3 vengono definiti come uscite e i pin RB4-RB6 sono invece definiti come ingressi; vengono inoltre abilitati via software i pull-up interni del PORTB. Successivamente vengono dichiarate le variabili di programma ed in particolare, per la memorizzazione del codice letto da input, vengono riservati 4 byte della memoria ram. La parte principale del firmware è quella relativa alla procedura di scansione della tastiera a matrice.

La scansione viene eseguita selezionando una riga alla volta ed andando poi ad interrogare in successione lo stato delle colonne. Il firmware implementa anche delle procedure antibounce al fine di sopprimere l'effetto dei rimbalzi dei contatti in fase di lettura, ed inoltre, se nessun tasto è premuto per un certo tempo, manda in "sleep" il controllore: con questa caratterista di funzionamento si attiva lo stato di basso consumo, con tutti i circuiti interni che assorbono il minimo di corrente. I dati letti dalla tastiera, vengono memorizzati temporaneamente nella RAM e vengono confrontati da una apposita subroutine, con i valori scritti nella EEPROM; nel firmware sono previste due possibilità operative di funzionamento: "no match" (ovvero codice letto diverso da codice memorizzato), che fa entrare il controllore in un loop infinito di lettura (da cui si esce solo quando viene immessa la corretta sequenza di attivazione) e "match" se invece viene riscontrata la corrispondenza, cifra per cifra, tra codice letto e codice memorizzato.

In quest'ultimo caso, a seconda del carattere di conferma ricevuto (* o #), viene rispettivamente attivata la chiave, oppu-

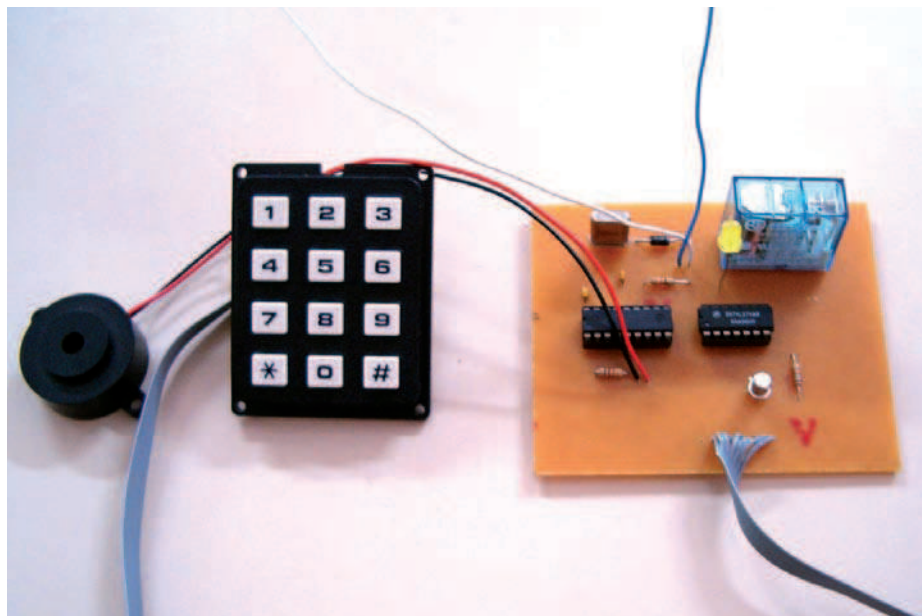



Figura 9: la chiave elettronica assemblata sul circuito stampato.

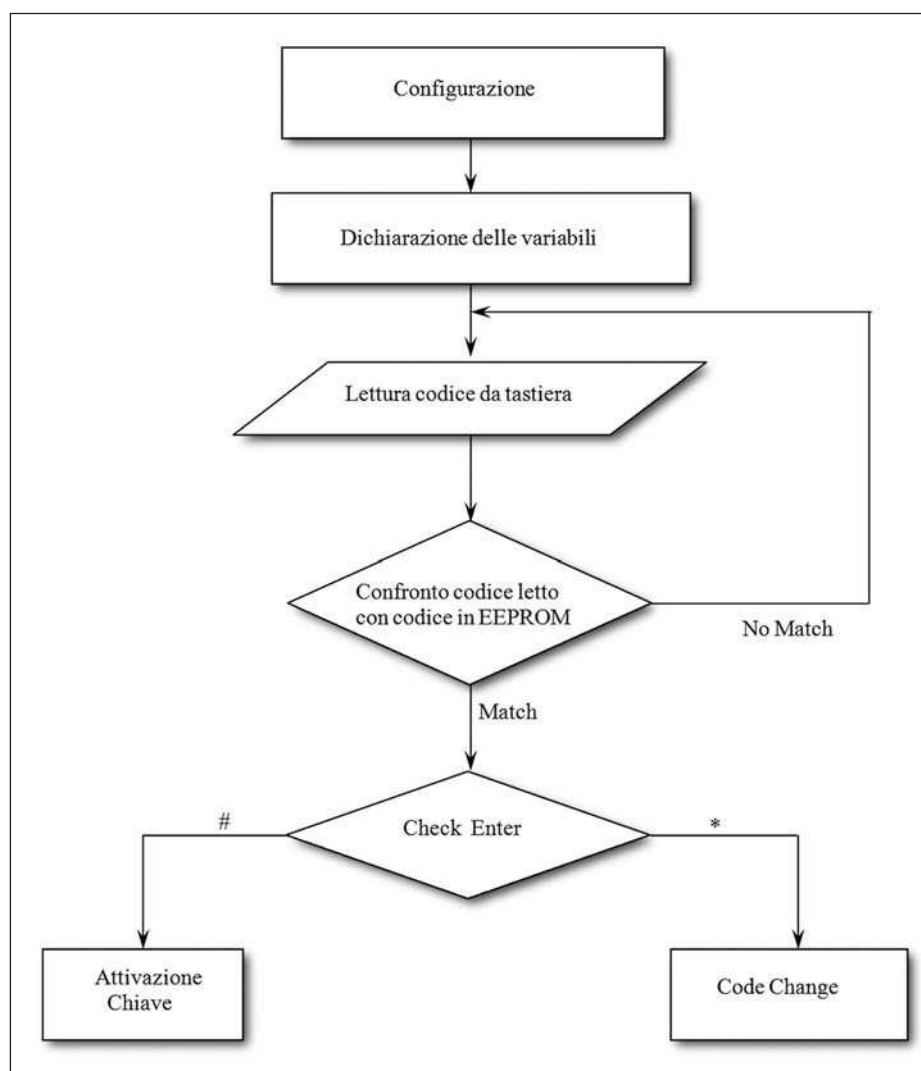



Figura 10: diagramma di flusso del firmware di controllo.

re la modalità cambio codice. Il codice predefinito alla prima attivazione del programma è “1234”. Una apposita subroutine si occupa invece dell’aggiornamento, ovvero della memorizzazione del nuovo codice di accesso, quando viene attivata dall’utente l’apposita sequenza per il cambio chiave: in questo caso, quindi, il nuovo codice numerico di attivazione, viene scritto dal firmware all’interno della memoria EEPROM.

La memoria dati EEPROM del PIC16F84 ha una capacità di 64 byte e ben si presta per essere impiegata per la memorizzazione di valori numerici che devono essere conservati anche se l’integrato non è alimentato: infatti ciascuna locazione della memoria EEPROM dati può essere letta o scritta durante il normale funzionamento del PIC senza necessità di utilizzare il circuito di programmazione del microcontrollore.

La memorizzazione dei dati nella EEPROM è fondamentale nella nostra applicazione, per salvare il codice di attivazione, che necessita di essere conservato anche se, per qualsiasi motivo, viene a mancare tensione all’integrato.

Il costruttore garantisce che sulla memoria EEPROM dati si possono effettuare fino a 10 milioni di scritture/letture e che i dati possono essere mantenuti in memoria per oltre 40 anni. Le procedure software relative alla scrittura e alla lettura nella EEPROM dati si basano essenzialmente sull’utilizzo del registro di controllo EECON1 insieme all’altro registro EECON2, gestito dal PIC.

Tali operazioni richiedono anche l’utilizzo di due registri speciali per la memorizzazione dei dati: il registro EEADR, allocato all’indirizzo 09H del File Register, che contiene l’indirizzo di una delle 64 locazioni di memoria EEPROM su cui operare; ed il registro EEDATA, allocato all’indirizzo 08H del File Register, che è impiegato per inviare alla EEPROM dati il byte da scrivere, oppure per ricevere un byte in una operazione di lettura dalla EEPROM dati. Il firmware di controllo è disponibile nella sezione download del sito www.farelettronica.com. ❏

CODICE MIP 2756737

Telecamere SPEED DOME

Speed Dome 22x con Intelligent Autotracking Control

Speed Dome di ultima generazione, a colori, con un eccezionale rapporto prezzo/prestazioni. La compattezza e la leggerezza rendono questa telecamera facilmente installabile ovunque. Sensore 1/4" Color Super HAD CCD da 480 linee TV, sensibilità 0,3 lux, zoom ottico 22x, veloce sistema di spostamento Pan e Tilt con 360° di rotazione orizzontale e 90° di inclinazione verticale, autofocus, alimentazione 12Vdc/1,5A.
La funzione Intelligent Autotracking Control consente di seguire automaticamente un target (oggetto/persona) all'interno di un'area preimpostata tenendolo al centro dell'inquadratura per tutto il tempo di permanenza nell'area. La telecamera può essere controllata mediante la console FR364 in grado di gestire fino a 64 unità. Il controller è munito di joystick di precisione, per comandare gli spostamenti, e di pulsanti per l'impostazione dello zoom, delle inquadrature predefinite (preset) e di tutte le altre funzioni.

FR369 € 580,00

Speed Dome professionale 3x a colori

Compatta Speed Dome da soffito, per interno, ideale complemento per i sistemi di videosorveglianza che utilizzano il controller FR364 e la Speed Dome FR363. Caratterizzata da una risoluzione di ben 480 linee TV, dispone di ottica zoom 4-9 millimetri per un'apertura angolare compresa tra 40° e 90°.
Sensore CCD 1/3"; sensibilità 0,3 lux; luminosità, contrasto, tonalità e nitidezza regolabili; bilanciamento del bianco ATW; effetto immagine "mirror" e funzione autofocus. Interfaccia grafica OSD per una immediata impostazione delle funzioni, zoom ottico 3x, 16 preset, rotazione pan 360°, escursione tilt 90°. Tensione di alimentazione 12Vdc, dimensioni Ø 134 x 122 mm, peso 1,2 kg.

Antivandalo 520 linee TV con LED IR

new

Telecamera a colori per installazione a soffitto dotata di un'eccellente risoluzione (ben 520 linee TV) e di elevata sensibilità (0 lux con LED IR attivi) grazie all'utilizzo di un elemento sensibile Sony CCD da 1/3".
Il contenitore in alluminio pressofuso (con grado di protezione IP67) conferisce alla telecamera doti di eccezionale robustezza a prova di atti vandalici. È dotata di 24 LED all'infrarosso, ad attivazione automatica, che permettono una visione ottimale anche in condizioni di scarsa luminosità. Particolarmente adatta per impianti di videosorveglianza in luoghi pubblici, palestre, corridoi, scuole, ecc. Specifiche: 752 X 582 pixel; ottica 3,6mm; otturatore elettronico 1/50 – 1/100.000s; alimentazione 12Vdc/250mA (max); portata LED 8-12m; dimensioni Ø94 x 73 x 55mm.

FR373 € 130,00

Antivandalo 540 linee TV con LED IR

new

Telecamera dome a colori da esterno (IP67) con obiettivo varifocal 3,5-8 mm, dotata di una eccellente risoluzione (ben 540 linee TV) e di elevata sensibilità (0 lux con LED IR attivi) grazie all'utilizzo di un elemento sensibile Sony CCD da 1/3". Il contenitore in alluminio pressofuso, completo di staffa, conferisce alla telecamera doti di eccezionale robustezza a prova di atti vandalici.
Dispone di 17 LED all'infrarosso ad attivazione automatica che permettono una visione ottimale anche in condizioni di luce critica. Possibilità di regolare la compensazione controluce, il controllo del guadagno, l'otturatore elettronico e le altre funzioni dell'immagine. Alimentazione 12V/250mA (alimentatore non incluso).

FR378 € 169,00

Prezzi IVA inclusa.

... e telecamere DOME

Con sensore CCD B/N

Dispone di sensore CCD 1/3" B/N e di contenitore a cupola adatto per montaggio a soffitto. Sistema standard CCIR, 400 linee TV, risoluzione 290.000 pixel; sensibilità 0,25 lux; otturatore elettronico Auto iris (velocità 1/60 ÷ 1/100.000); ottica f=3,6mm / F 2.0; alimentazione 12Vdc; dimensioni Ø 87 x 58mm; peso 96g.

FR155 € 60,00

Con sensore CCD Sharp a colori

Telecamera completa di adattatore di rete caratterizzata da un elemento sensibile CCD 1/4" Sharp con ottica f =6,0mm / F1.2. Sistema video PAL, 420 linee TV; pixel effettivi 512(L) x 582(H); sensibilità 1 lux; apertura angolare 60°; uscita video 75 ohm 1Vpp (BNC); alimentazione 12Vdc/100mA; peso 300g; dimensioni Ø 113 x 80mm.

CP316 € 52,00

Con sensore CMOS 1/3" a colori

Telecamera adatta per installazioni a soffitto, completa di adattatore di rete. Elemento sensibile CMOS 1/3" Omnivision, risoluzione 380 linee TV; pixel effettivi 628(L) x 582(H); sensibilità 3 lux; sistema video PAL; uscita video 75 ohm 1Vpp (BNC); ottica f=6,0mm/F1.2; apertura angolare 60°; alimentazione 12Vdc/100mA; peso 300g; dimensioni 113 x 80mm.

CP317 € 38,00

Con sensore CCD Sharp a colori e IR

Può essere utilizzata per effettuare riprese in condizioni di buio assoluto (10m max.) grazie ai 12 LED IR, con attivazione automatica integrati. È dotata di sensore CCD da 1/4" con sensibilità di 1 lux (0 lux, IR ON) e risoluzione di 420 linee TV. Pixel 512(L) x 582(H); sistema video PAL; uscita video 75 ohm 1Vpp (BNC); ottica f=6,0mm/F1.2; apertura angolare 60°; alimentazione 12Vdc/100mA (120mA, IR ON); peso 300g; dimensioni 113x80mm.

CP298 € 48,00

CODICE MIP 2753547

di ROBERTO PRESTIANNI

CONTROLLO Relè VIA USB

Semplice scheda con quattro relè di potenza da pilotare direttamente via USB. Impiegando il noto FT232RL, studiamo la possibilità di interfacciare i nostri circuiti al PC in modo semplice e veloce

Il progetto che andiamo a proporvi consente il controllo di quattro relè di potenza direttamente dal proprio PC. L'interfacciamento è realizzato da Hyper Terminal, un noto e diffuso programma per il collegamento del computer con sistemi remoti di vario tipo. La comunicazione impiegata da questo terminale software avviene selezionando la porta seriale COM generata virtualmente dal convertitore FT232RL, un IC che viene connesso direttamente ad una delle porte USB del computer. Anche in questo caso la logica del funzionamento è affidata ad un microcontrollore Microchip della serie PIC16 il quale, interfacciandosi al convertitore tramite UART (Universal Asynchronous Receiver-Transmitter), è in grado di stabilire una comunicazione bidirezionale col PC per la ricezione dei comandi da esso impartiti. Il firmware che gira al suo interno realizza, assieme ad Hyper Terminal, un interfacciamento "verboso" con messaggi testuali che guidano passo passo l'utente nel controllo della scheda e quindi delle sue uscite. In questo modo, inserendo semplici parole chiave specificanti il tipo di comando da impartire e l'uscita desiderata, sarà possibile controllare lo stato di quest'ultima. Il progetto che ci accingiamo a descrivervi ha sicuramente una sua utilità pratica, qualora si intenda pilotare alcuni apparecchi elettrici (anche a 220Vac) ma, indubbiamente, ha un suo riscontro anche a livello didattico, dal momento che il circuito rappresenta un esempio

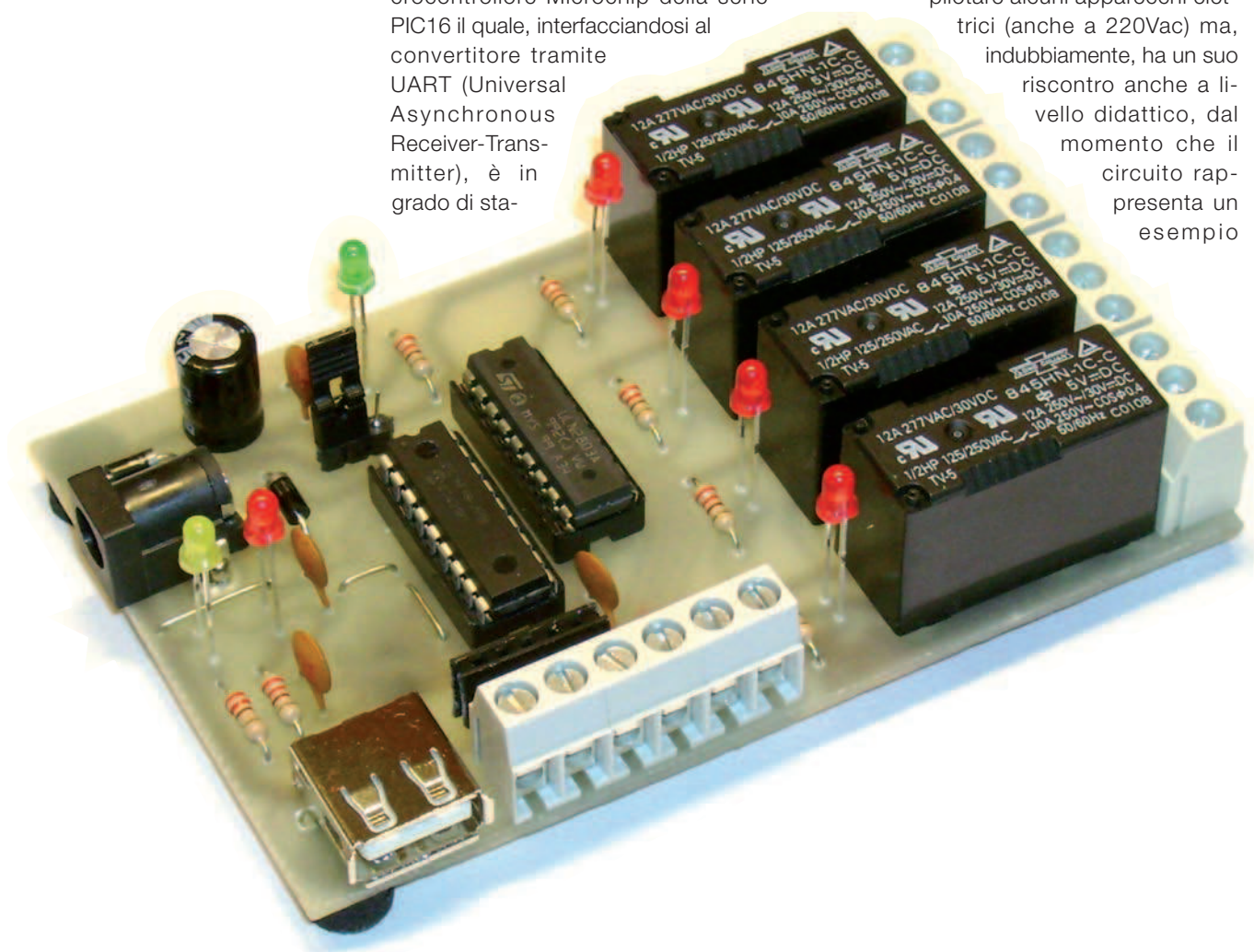
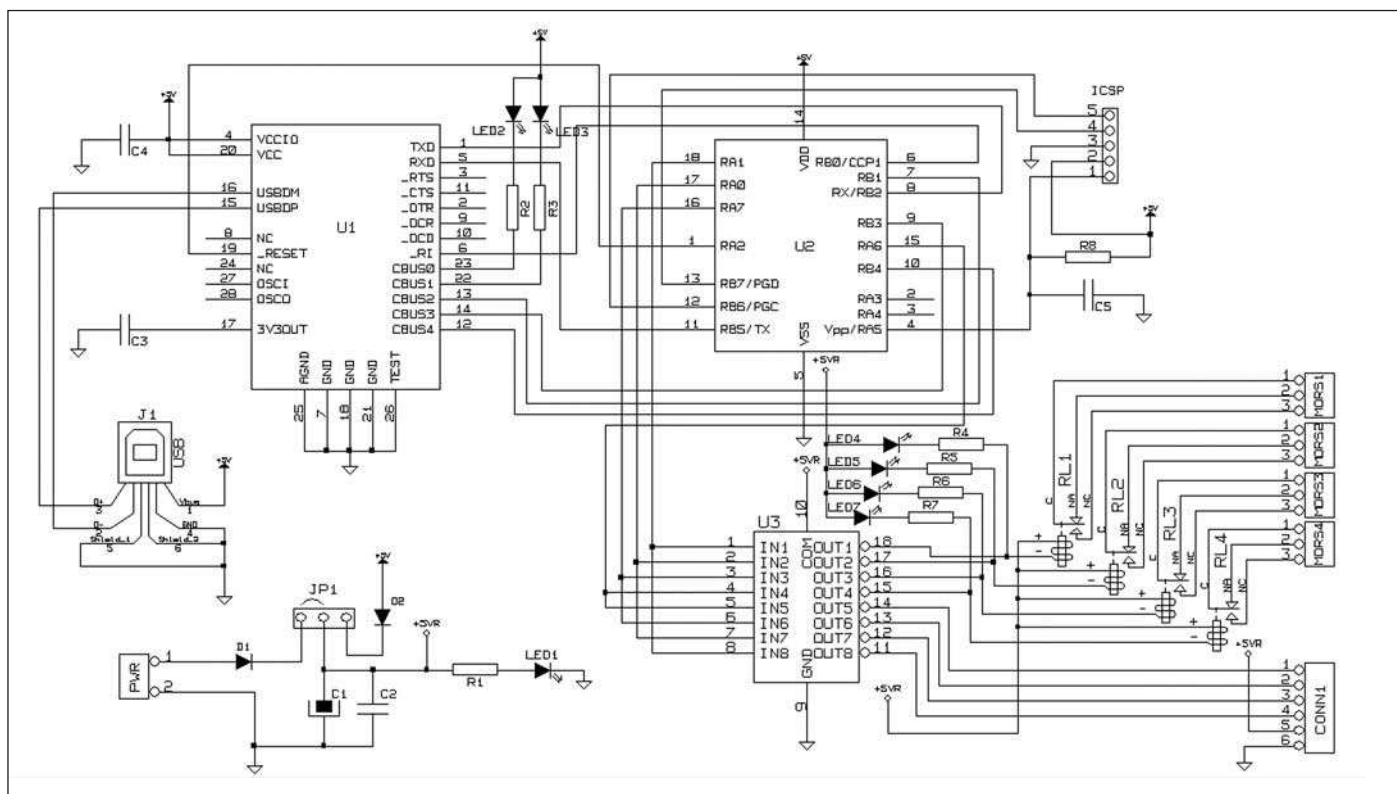



Figura 1: schema elettrico della scheda relè.

di interfacciamento tra un microcontrollore ed il convertitore USB/UART FT232RL. Si vedrà inoltre come sia possibile personalizzare sia la connessione USB implementata da questo integrato, sia la configurazione di alcune sue linee di I/O tramite un'apposita applicazione messa a disposizione dalla FTDI, la casa elettronica che produce questo chip.

SCHEMA ELETTRICO

In **figura 1** viene mostrato lo schema elettrico della scheda. Notate per prima cosa i tre circuiti integrati U1-U2-U3, rispettivamente il convertitore a montaggio superficiale FT232RL (in contenitore SSOP da 28 pin), il microcontrollore PIC16F88 e l'ottuplo Darlington (entrambi in contenitore PDIP da 18 pin). La sezione logica del dispositivo viene alimentata direttamente dalla porta USB che, lo ricordiamo, è in grado di fornire una tensione stabilizzata +5V e di erogare una corrente massima di 500mA. La parte di potenza invece, quella che include i Darlington (U3), i relè (RL1/2/3/4) e i relativi led di segnalazione (LED4/5/6/7), può essere alimentata sia da una fonte esterna applicata al plug (PWR), sia dalla stessa porta USB, dal momento che il consumo massimo (con i quattro relè accesi) è ben al di sotto dei 500mA. L'una o l'altra fonte di alimentazione viene selezionata posizionando su JP1 il relativo ponticello in un modo o nell'altro. Questa soluzione è stata pensata in primo luogo per non caricare eccessivamente la porta USB, nel caso in cui non sia possibile prelevare da questa la corrente necessaria all'intera scheda, e in secondo luogo per fornire un grado di libertà sulla scelta della tensione di alimentazione dei relè: infatti, dal momento che le uscite di U3 sono open collector, ci sarà consentito alimentare lo stadio di uscita con tensioni differenti dai 5V, a patto che si usino per i led resistenze di valore adeguato e non si superi la tensione massima consentita da U3 (50V). Uno dei motivi che potrebbe limitare la corrente massima prelevabile da una periferica da una delle porte USB del PC, potrebbe essere l'interposizione

LISTA COMPONENTI

R1-R7	**220 Ω1/4 W**	**Varie**
R8	**10 KΩ 1/4 W**	**Connettore USB femmina tipo A per cs**
C1	**220 µF 25 V elettrolitico**	**4 x morsettiere tripolari per cs**
C2-C5	**100 nF ceramico**	**Morsettiera 6 poli per cs**
D1,D2	**1N4007**	**Plug di alimentazione per cs**
LED1	**Led 3mm verde**	**Strip maschio 3 pin + jumper 2 pin**
LED2	**Led 3mm giallo**	**Strip femmina 5 pin**
LED3-LED7	**Led 3mm rosso**	**2 x Zoccolo per IC 9+9 pin**
U1	**FT232RL**	**Cavetto USB maschio/maschio tipo A**
U2	**PIC16F88-I/P**	**CS**
U3	**ULN2803A**	
RL1-RL4	**Relè 5V monoscambio 250V-10A (vedere testo)**	

Figura 2: pinout del microcontrollore PIC16F88.

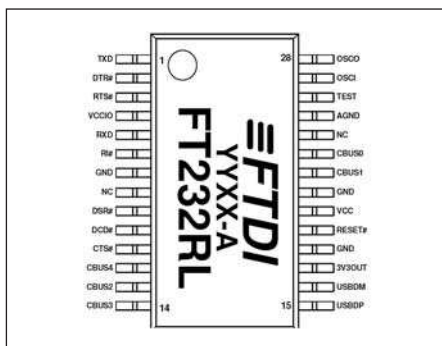



Figura 3: pinout del convertitore FT232RL.

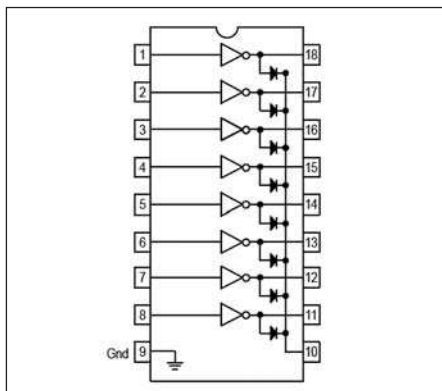



Figura 4: pinout dell'ottuplo Darlington ULN2803A.

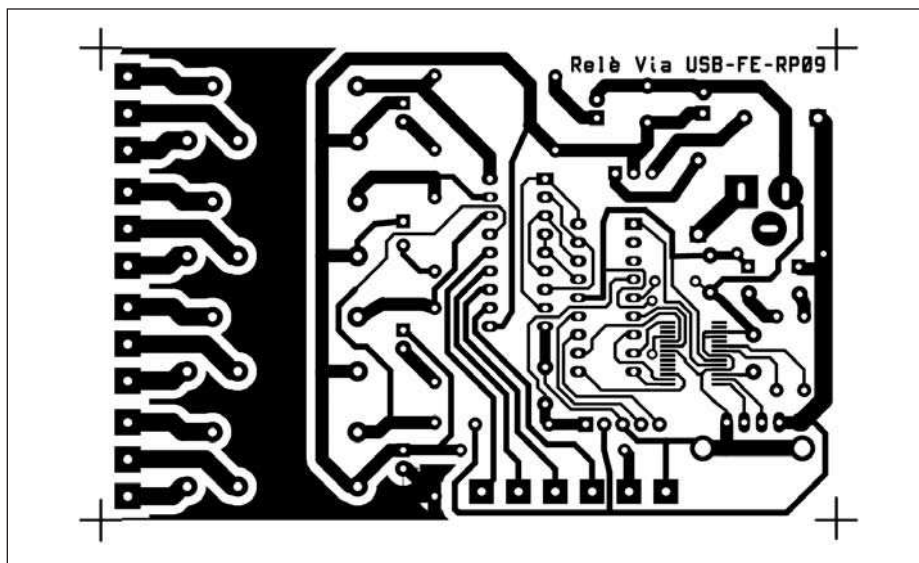



Figura 5: circuito stampato lato rame in scala 1:1 (10x6,5 cm).

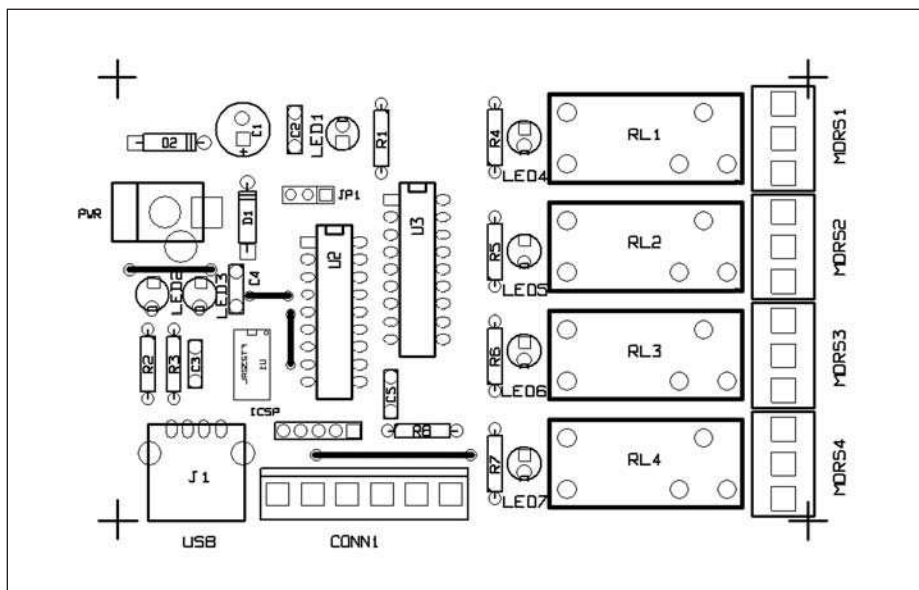



Figura 7: piano di montaggio dei componenti (U1 è visto in trasparenza).

di un HUB non autoalimentato. Quest'ultimo infatti, dal momento che serve molteplici porte USB, dovrà ripartire tra queste la corrente che preleverà dalla porta USB alla quale verrà connesso. Ritorniamo al convertitore U1: il connettore J1 risulta essergli connesso direttamente, sia dunque per l'alimentazione (pin 1 e 4 rispettivamente +5V e GND) che per le linee seriali D- e D+ (pin 2 e 3). Quest'ultimi fanno capo ai piedini 16 (USBDM) e 15 (USBDP). Oltre alle diverse linee di alimentazione (VCC e GND), l'integrato U1 dispone anche delle nove linee dedicate alla comunicazione seriale secondo lo standard RS232, ovvero: TXD, RXD, RTS, CTS, DTR, DCR, DCD, RI e GND. Basterebbe la sola interposizione di un convertitore di livello per RS232 tra queste linee e un connettore DIN a 9 poli, per realizzare appieno un convertitore USB/RS232, come quelli che si trovano nei negozi di informatica. Nel contesto della scheda relè che stiamo descrivendo invece, si impiegano semplicemente le linee di trasmissione e ricezione TXD (pin 1) ed RXD (pin 5), le quali consentono l'interfacciamento asincrono col modulo UART integrato al PIC U2 (linee RX e TX - pin 8 e 11). Oltre a queste otto linee dedicate, il convertitore dispone di un BUS da 5 linee (CBUS0/1/2/3/4) configurabili a piacere tramite l'applicazione MProg (oggi giunta alla versione 3.5) realizzata dalla stessa FTDI Chip. Ad esempio, osservate come CBUS0 e CBUS1 pilotino direttamente i due led di segnalazione LED2 e LED3: il primo si accende ogni volta che un carattere ASCII viene emesso, mentre il secondo quando invece un carattere sarà ricevuto. Le altre linee del BUS (CBUS2/3/4) sono tutte collegate ad altrettanti input del microcontrollore. Questo accorgimento crea una risorsa futura, così da lasciare spazio ai lettori ad

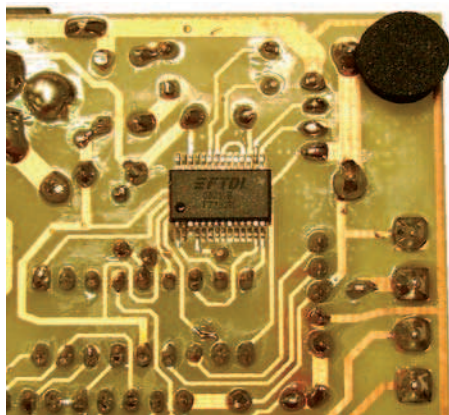



Figura 6: risultato della saldatura del convertitore FT232RL sul lato rame del cs.

eventuali upgrade del sistema: ad esempio la linea RI di U1 può essere impiegata per il wake-up della scheda. Nel seguito dell'articolo verrà mostrato come eseguire queste ed altre configurazioni via USB. Ogni volta che colleghiamo la scheda alla porta USB del PC, questa stessa viene alimentata ed il PIC immediatamente resettato grazie alla rete RC R8-C5, connessa a Vpp/MCLR (pin 4). Una volta avviato il firmware anche il convertitore viene resettato immediatamente tramite la linea di out RA2 (pin 2), sulla quale viaggerà un impulso basso di durata di circa 10mSec. Per facilitare la personalizzazione del firmware è stato inserito un connettore strip femmina (ICSP) per la connessione al programmatore, il quale riporta l'esatta piedinatura del PICKit2 (1-Vpp, 2-Vcc, 3-GND, 4-PGD, 5-PGC). Per concludere la descrizione dello schema elettrico, si osservino le connessioni delle linee di output RA1/RA0/RA7/RA6 con le quali vengono pilotati i quattro relè (rispettivamente RL1-RL2-RL3-RL4) tramite i Darlington di U3. Queste linee sono inoltre gemellate in uscita da U3 e connesse alla morsettiera CONN1 assieme all'alimentazione. Questo accorgimento tornerà utile nel caso si intenda impiegare le stesse quattro uscite senza la necessità dei contatti puliti dei relè.

REALIZZAZIONE PRATICA

Realizzate il master del circuito stampato utilizzando l'immagine del lato rame in scala 1:1 di **figura 5**, e create così il PCB della scheda per fotoincisione. Ponete particolare attenzione all'area della pia-

Figura 8: risultato dell'assemblaggio dopo la saldatura dei componenti.

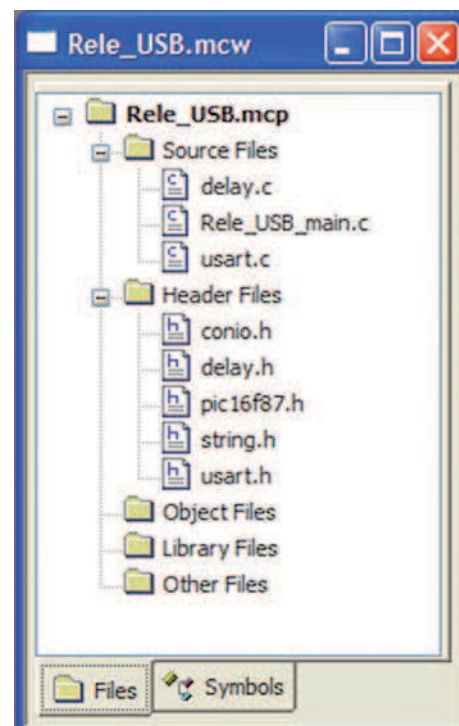



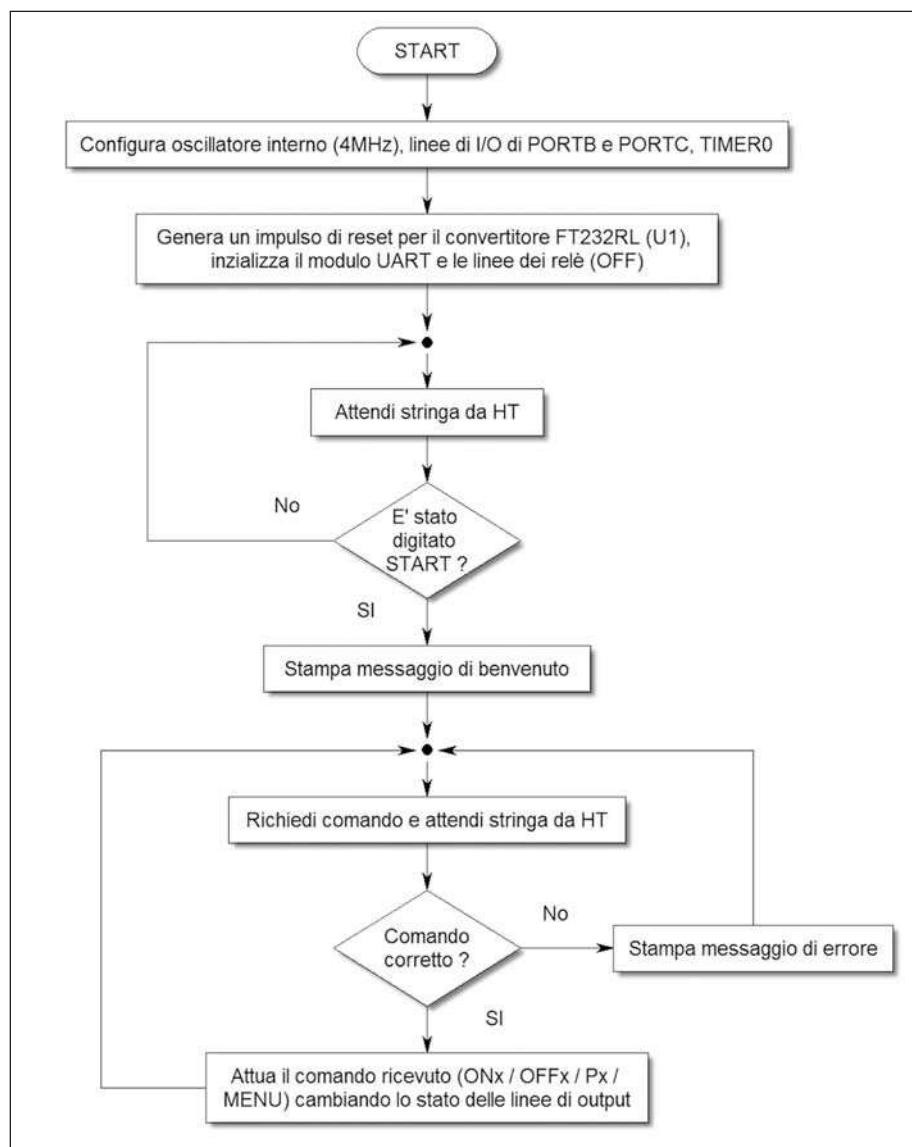
Figura 9: struttura del project MPLAB.

strina in cui va montato il circuito integrato smd, dal momento che i relativi pad e le piste di collegamento possono risultare molto vicine tra loro. In merito alla foratura si consiglia di usare preliminarmente una punta da trapano da 0,8mm per tutti i pad , e poi passare a quella da 0,9mm per i diodi D1 e D2, lo strip maschio JP1 e lo strip femmina per l'ICSP, e ancora una punta da 1,2mm per i pad di tutte le morsettiere e i quattro relè, una da 2,2mm per i terminali laterali del connettore USB e da 2,5mm per quelli del plug di alimentazione. Fatto questo passate alla saldatura di tutti i componenti: si inizia con la parte più delicata di tutto il processo di realizzazione della scheda, ovvero la saldatura sul lato rame di U1. In **figura 6** è mostrato il risultato di questa prima saldatura e come deve essere orientato il circuito integrato (il pin 1 – contrassegnato dal pallino impresso sul contenitore di U1 – deve essere orientato in basso e a sinistra). Proseguite con i quattro ponticelli mostrati in **figura 7**, le otto resistenze e i due diodi, i due zoccoletti da 18 pin (9+9) per U2 e U3, e poi il connettore USB di tipo A, lo strip maschio da tre pin (JP1) e quello femmina da 5 pin (ICSP).

Figura 10: Flow-chart della main().

LISTATO 1

```
//Esecuzione comando appena letto
//*** Attivazione relè (ON) *****
   if(strcmp(buffer, “ON1”)     == 0 || strcmp(buffer, “on1”)    == 0) OUT1=1;
else if(strcmp(buffer, “ON2”)    == 0 || strcmp(buffer, “on2”)    == 0) OUT2=1;
else if(strcmp(buffer, “ON3”)    == 0 || strcmp(buffer, “on3”)    == 0) OUT3=1;
else if(strcmp(buffer, “ON4”)    == 0 || strcmp(buffer, “on4”)    == 0) OUT4=1;
else if(strcmp(buffer, “ONALL”)  == 0 || strcmp(buffer, “onall”)  == 0)
                                 {OUT1=1;OUT2=1;OUT3=1;OUT4=1;}
//*** Disattivazione relè (OFF) *****
else if(strcmp(buffer, “OFF1”)   == 0 || strcmp(buffer, “off1”)   == 0) OUT1=0;
else if(strcmp(buffer, “OFF2”)   == 0 || strcmp(buffer, “off2”)   == 0) OUT2=0;
else if(strcmp(buffer, “OFF3”)   == 0 || strcmp(buffer, “off3”)   == 0) OUT3=0;
else if(strcmp(buffer, “OFF4”)   == 0 || strcmp(buffer, “off4”)   == 0) OUT4=0;
else if(strcmp(buffer, “OFFALL”) == 0 || strcmp(buffer, “offall”) == 0
                                 {OUT1=0;OUT2=0;OUT3=0;OUT4=0;}
```

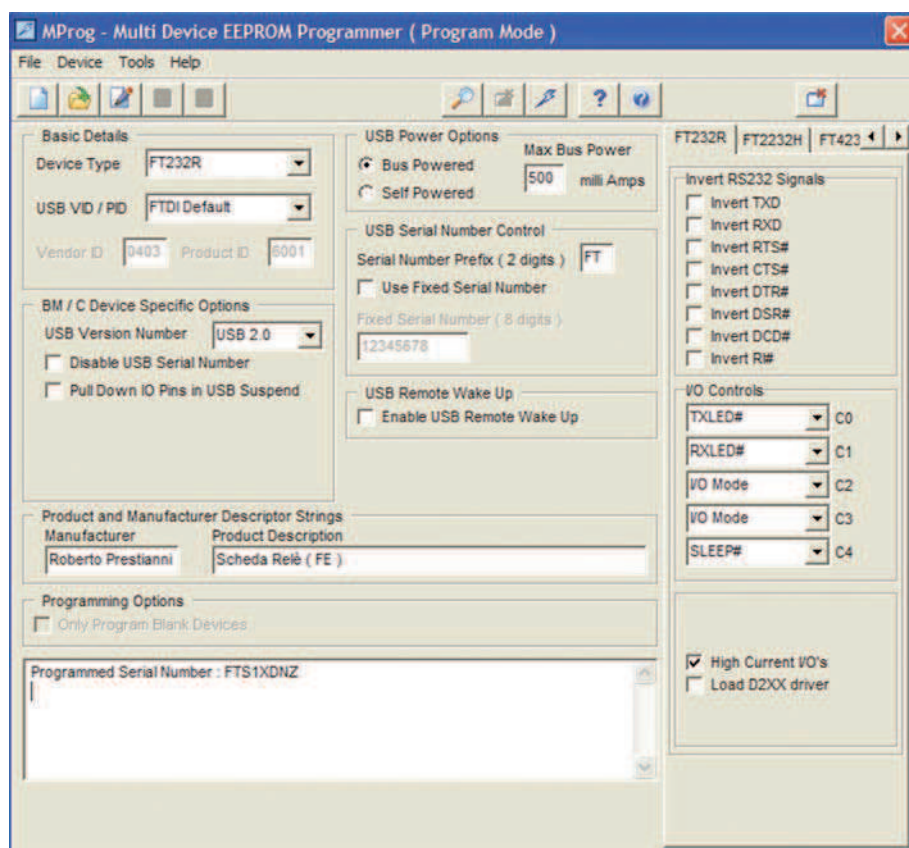



Figura 11: programma di configurazione del convertitore FT232RL.

A seguire i quattro condensatori ceramici, il plug di alimentazione, il condensatore elettrolitico, le cinque morsettiere e, per concludere, i quattro relè e i sette led da orientare tutti con l’anodo dal lato opposto alla porta USB, eccetto LD1, il led di alimentazione. Controllate tutte le saldature e assicuratevi che non vi siano ponticelli accidentali, in modo particolare tra i pad di U1. Inoltre verificate la giusta connessione dei pin del connettore USB con il convertitore. Con un ponticello selezionate l’alimentazione della sezione di potenza della scheda, inserendolo sullo strip JP1: se lo disponete a sinistra (verso il plug) alimenterete tutto con i 5V della porta USB, se posto a destra invece alimenterete i relè con l’alimentazione presente al plug (il terminale positivo stà al centro). A questo proposito, come già accennato in precedenza, siete liberi di utilizzare relè con tensione diversa dai 5V, l’importante è non superare la massima tensione ammissibile dalle uscite dei Darlington (circa 50V) e dimensionare adeguatamente le resistenze limitatrici dei quattro led LED4/5/6/7. Conviene a questo punto collegarsi al sito della FTDI e scaricare i driver del convertitore FT232R e il programma di con-

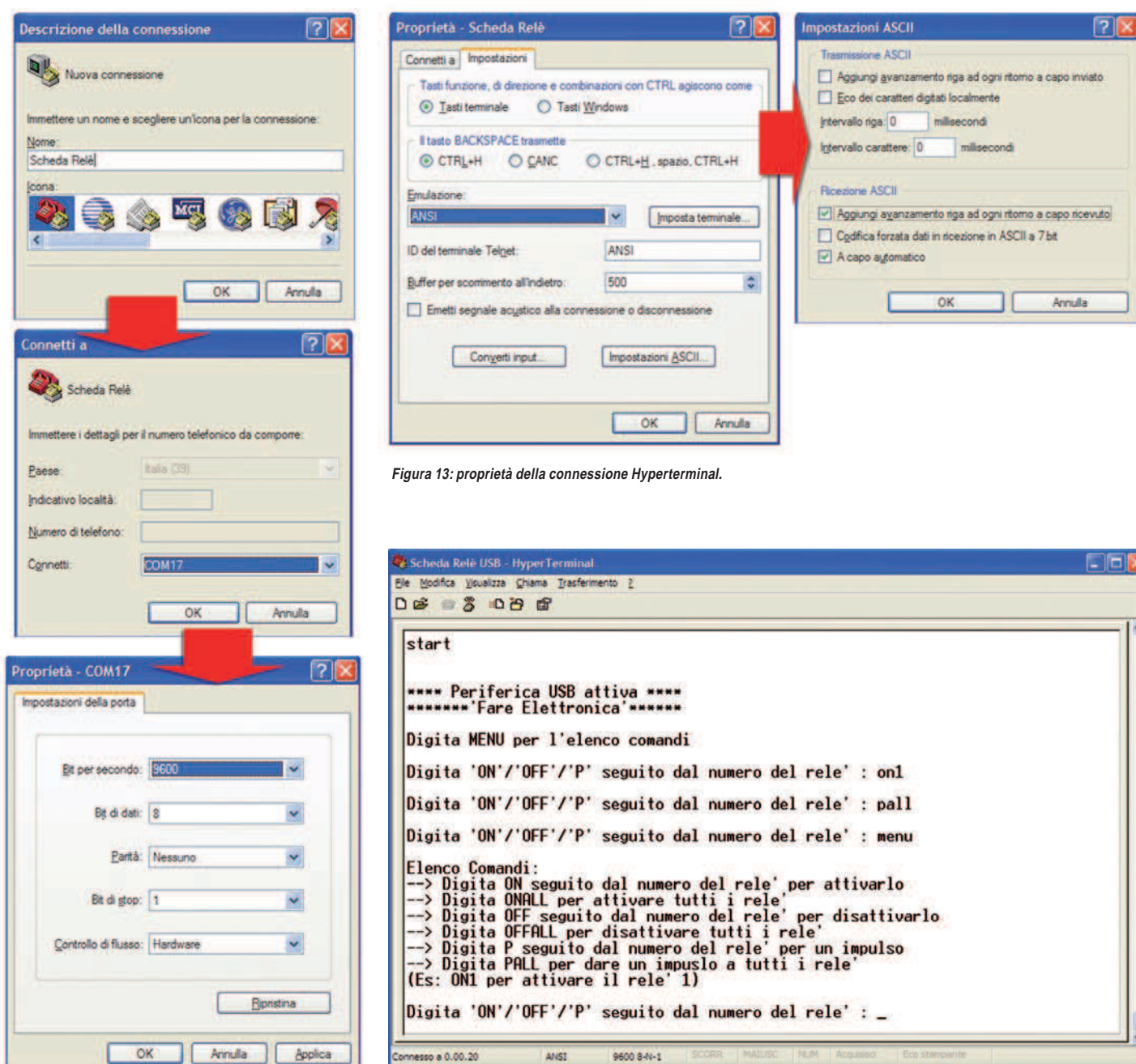



Figura 12: sequenza di configurazione iniziale di Hyperterminal.

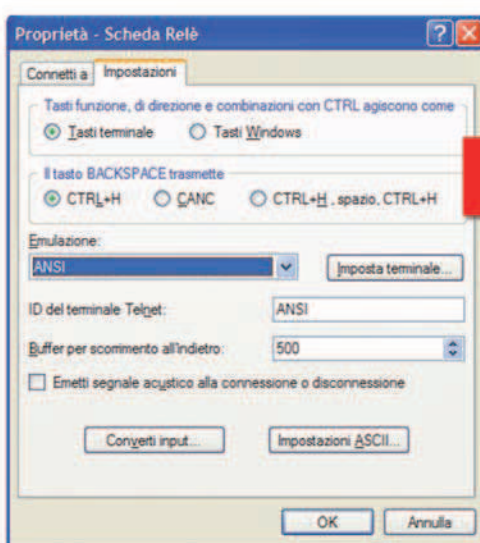



Figura 13: proprietà della connessione Hyperterminal.

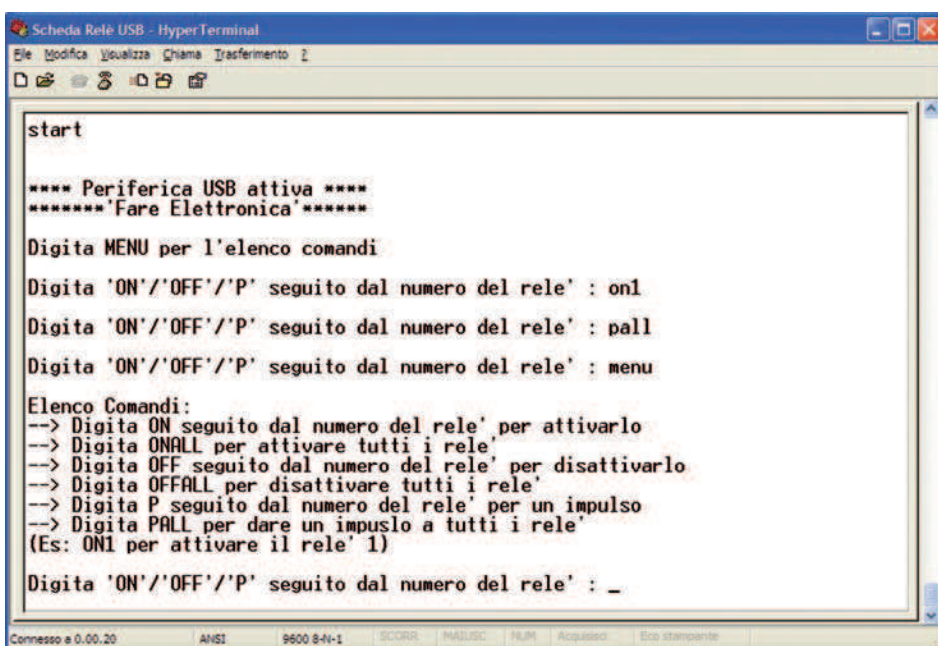
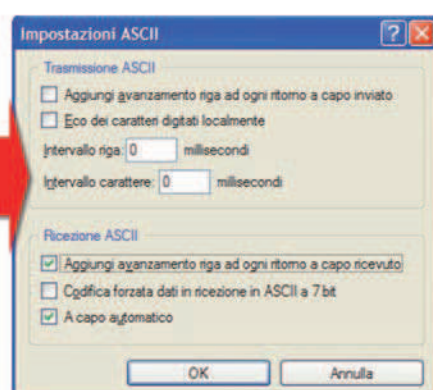



Figura 14: esempio di interazione tramite Hyperterminal.

figurazione MProg: il driver, oggi giunto alla versione 2.04.16 (per Windows XP), lo trovate all'indirizzo *http://www.ftdichip.com/Drivers/VCP.*htm, mentre MProg lo trovate come *utility* alla pagina *http://www.ftdichip.com/Resources/Utilities.htm#MProg* in versione zippata (MProg3.5.zip). Un primo test elettrico può essere fatto collegando la scheda ad una porta USB del PC con un cavetto USB maschio/maschio. Nel momento stesso che effettuate la connessione il led verde dell'alimentazione dovrà accendersi, caso contrario scollegate il cavetto e ricontrollate tutto. Leggete la tensione tra i pin 14 e 5 dello zoccoletto di U2: qui dovrete leggere circa 5V, la stessa tensione prelevata dal bus USB. Una volta eseguita la connessione, il sistema operativo tenterà di installare i driver della periferica, quindi, a quel punto, dovrete indicargli la cartella in cui avete salvato i driver dell'FT232R. Terminata l'installazione la periferica sarà etichettata come "USB Serial Converter". Scollegate il cavo e inserite i due intergrati U2 ed U3 sui rispettivi zoccoletti, entrambi da orientare con la tacca di riferimento (pin 1) verso il led verde. A questo punto non rimane che programmare il microcontrollore: scaricate dal sito della rivista il file *Rele_USB.hex*, il firmware per il PIC, e caricatelo sull'integrato. Se usate il programmatore PICKit2 fate in modo che la linea 1 venga connessa al polo estremo del connettore ICSP rivolto verso i relè.

IL FIRMWARE

In **figura 9** e riportata la struttura del project stilato in linguaggio ANSI C in ambiente MPLAB:
qui sono elencati i file .c, ovvero delay.c contenente alcune funzioni di ritardo impieganti il TIMER0, Rele_USB_main.c nella quale viene definita la main e quindi il codice principale del firmware, e usart.c che definisce le funzioni di scambio di caratteri ASCII quali getch(), putch(), getche() rispettivamente per l'acquisizione, l'emissione e l'eco di un singolo carattere. In questo progetto quest'ultime funzioni non sono impiegate però, all'occorrenza, possono tornare utili. A seguire sono elencati tutti i file haders: conio.h che dichiara una serie di funzioni di I/O come la cputs() e la cgets utilizzate dalla main rispettivamente per emettere o ricevere un'intera stringa, delay.h è il file header della delay.c, pic16f87.h che definisce i nomi di registri, flag ed altro relativamente al PIC16F88, string.h definisce alcune funzioni per la manipolazione delle stringhe, e usart.h che definisce le proprietà della connessione UART (come il baudrate a 9600), e la funzione init_comms() di inizializzazione dello stesso modulo seriale. Il codice della Rele_USB_main.c ha inizio con la *Configuration Bits*, la dichiarazione della variabile array buffer[], usata per la memorizzazione di tutte le stringhe, la definizione dei nomi delle linee di I/O del microcontrollore, e i prototipi delle funzioni che realizzano i ritardi. Pe meglio comprendere il funzionamento del firmware, si studi il semplice flow-chart di **figura 10**. Il **listato 1** mostra una parte del codice che riconosce il comando impartito dall'utente tramite HyperTerminal, e lo esegue pilotando opportunamente le uscite OUTx relative ai relè, o stampando lo shell dei comandi.

CONFIGURARE IL CONVERTITORE

Per prima cosa aprite MProg cliccando sulla relativa icona che trovate dentro la cartella precedentemente scaricata e scompattata (MProg 3.5 Release), e connettete nuovamente la scheda alla porta USB. Dal menù "Device" cliccate su "Scan" dimodochè il programma individui la periferica, o meglio il convertitore FT232R. Adesso, cliccando sull'icona della cartella appena sotto "Device", caricate il file *SchedaRele-FE.ept* che trovate sul sito della rivista. Vedrete apparire tutte le configurazioni scelte per l'interfacciamento della scheda Relè: USB 2.0, nome dell'autore e della periferica "*Scheda Relè (FE)*" (la stessa che troverete in *Gestione periferiche Controller USB (Universal serial bus) USB Serial Converter* di Windows XP), la selezione dell'alimentazione dal bus ("Bus Powered") e della corrente massima prelevabile dalla scheda (in questo caso 500mA) e le configurazioni delle linee C0/1/2/3/4 (vedere **figura 11**). Le prime due, presenti rispettivamente col nome di CBUS0 e CBUS1 del pinout dell'FT232RL, sono configurate per comandare LED2 (giallo) e LED3 (verde). Il primo (TXLED) si accende quando il convertitore riceve un carattere ASCII dal PC e lo trasmette al PIC. Il secondo invece (RXLED) si accende quando è U1 a ricevere un carattere dal PIC, che poi passerà al PC e quindi ad Hyper Terminal. Le altre linee di questo bus, sebbene siano tutte connesse ad altrettante linee di input del microcontrollore U2, in questo progetto non sono utilizzate. Ritornando alla configurazione della scheda, una volta caricato il file *.ept* non dovrete far altro che cliccare sull'icona del fulmine affinché MProg vada a scrivere la EEPROM di U1 via USB, e salvare così le configurazioni volute. A questo punto la scheda è pronta all'uso; non rimane che configurare molto velocemente Hyper Terminal ed iniziare a pilotare la scheda. Consigliamo a questo punto di esplorare le diverse possibilità offerte da MProg per poter configurare il convertitore in base alle proprie esigenze. Per maggiori dettagli sulle configurazioni potrete consultare la guida messa a disposizione dallo stesso MProg, mentre se desiderate maggiori informazioni sul convertitore, come può essere impiegato, per vedere alcuni esempi nonché scaricare il relativo datasheet, potete consultare la pagina internet:
http://www.ftdichip.com/Products/FT232R.htm

PER *approfondire...*

http://www.ftdichip.com (Home page FTDI)
http://www.ftdichip.com/Products/FT232R.htm (FT232R)
http://www.ftdichip.com/Drivers/VCP.htm (Driver)
http://www.ftdichip.com/Resources/Utilities.htm#MProg (MProg)

HYPERTERMINAL

Questo semplice terminale di comunicazione e presente in tutti i sistemi operativi Windows. In XP dovreste trovarlo seguendo il percorso: *"Start" "Tutti i programmi"->"Accessori"->"Comunicazioni"->"Hyper Terminal"*.
La sequenza in **figura 12** mostra passo passo le configurazioni iniziali che dovrete effettuare una volta aperto Hyperterminal. Giunti all'interfaccia vera a e propria cliccate su proprietà e configurate secondo quanto mostrato in **figura 13**, e poi salvate dal menu "File" queste configurazioni, in modo da non doverle ripeterle in futuro, (sempre che non andiate a cambiare porta USB di connessione alla scheda: in questo caso bisognerà modificare il numero della porta COM). Ritornati alla schermata principale, cliccate sull'icona del telefono per avviare la connessione e poi digitate la parola chiave "start", così da avviare la comunicazione tra Hyper Terminal e la scheda. Questa parola viene letta dal PIC che, a quel punto, emetterà un messaggio di benvenuto e si metterà in attesa di uno dei seguenti comandi:
- ONx: *ACCENSIONE Relè*
- OFFx: *SPEGNIMENTO Relè*
- Px: *PULSE Relè – Accensione ad impulso (circa 0,5 Sec)*
- ONALL / OFFALL / PALL: ON, OFF e P contemporanei per i quattro relè
- MENU: stampa riepilogo comandi.

NOTA: (la x specifica il relè 1-2-3-4)
Una volta terminata la sessione chiudete la comunicazione cliccando sull'icona del telefono con la cornetta alzata. Considerata la semplicità dell'interazione tra scheda e PC, nulla vieta di realizzare una piccola applicazione con un interfaccia virtuale fatta di pulsanti, interruttori ed altro che renda più immediato e intuitivo il controllo dei relè e non solo.
La cosa importante è che questa applicazione sia in grado di comunicare attraverso la porta COM virtuale creata dal convertitore U1. Ad esempio quando si pressa il pulsante di accensione del relè 1, l'applicazione dovrà subito inviare la stringa di comando relativa all'eccitazione di quel relè, ovvero quella codificata all'interno del firmware, in questo progetto "ON1" oppure "on1". ❑

CODICE MIP 2756723

l'angolo di Mr A.KEER

MULTIPLEXER

Con questo numero iniziamo l'analisi dei principali dispositivi combinatori, a cominciare dai Multiplexer; scopriremo i segreti e tutti i componenti resi disponibili sul mercato per realizzare questa importante struttura

Nelle precedenti puntate abbiamo avuto occasione di verificare che, per ogni variabile d'uscita, è possibile scrivere una equazione booleana, comunque caratterizzata dalla presenza di operazioni logiche (somme e/o prodotti) chiamate a combinare tra loro gli ingressi; la conoscenza delle porte logiche consente di realizzare concretamente ciascuna di esse, concorrendo alla realizzazione di qualunque progetto digitale. Per questa ragione questi dispositivi, realizzati esclusivamente con l'impiego di sole porte logiche, sono comunemente noti come macchine combinatorie; il loro funzionamento è comunque descritto dalla rispettiva Tabella di verità, cioè dalla struttura in grado di mostrare sinteticamente il modo con cui ciascuna variabile d'ingresso influenza lo stato logico delle uscite previste. Le macchine combinatorie sono piccoli gioielli di efficienza, specializzate in aritmetica (sommatori, comparatori, moltiplicatori) e nel controllo dei segnali (decoder, encoder, demultiplexer, multiplexer); avremo modo di indagare su ciascuno di essi, a cominciare da quest'ultimo.

GATING DEI SEGNALI LOGICI

Prima di affrontare lo studio dei multiplexer è interessante soffermare l'attenzione su un metodo molto ricorrente per controllare il passaggio dei segnali logici; in un progetto digitale la presenza delle porte logiche è legata alla funzione (somma o prodotto) ad esse richiesta, in accordo con la formula del circuito, ricavata dalla tabella di verità del progetto stesso. Stando così le cose può sem-

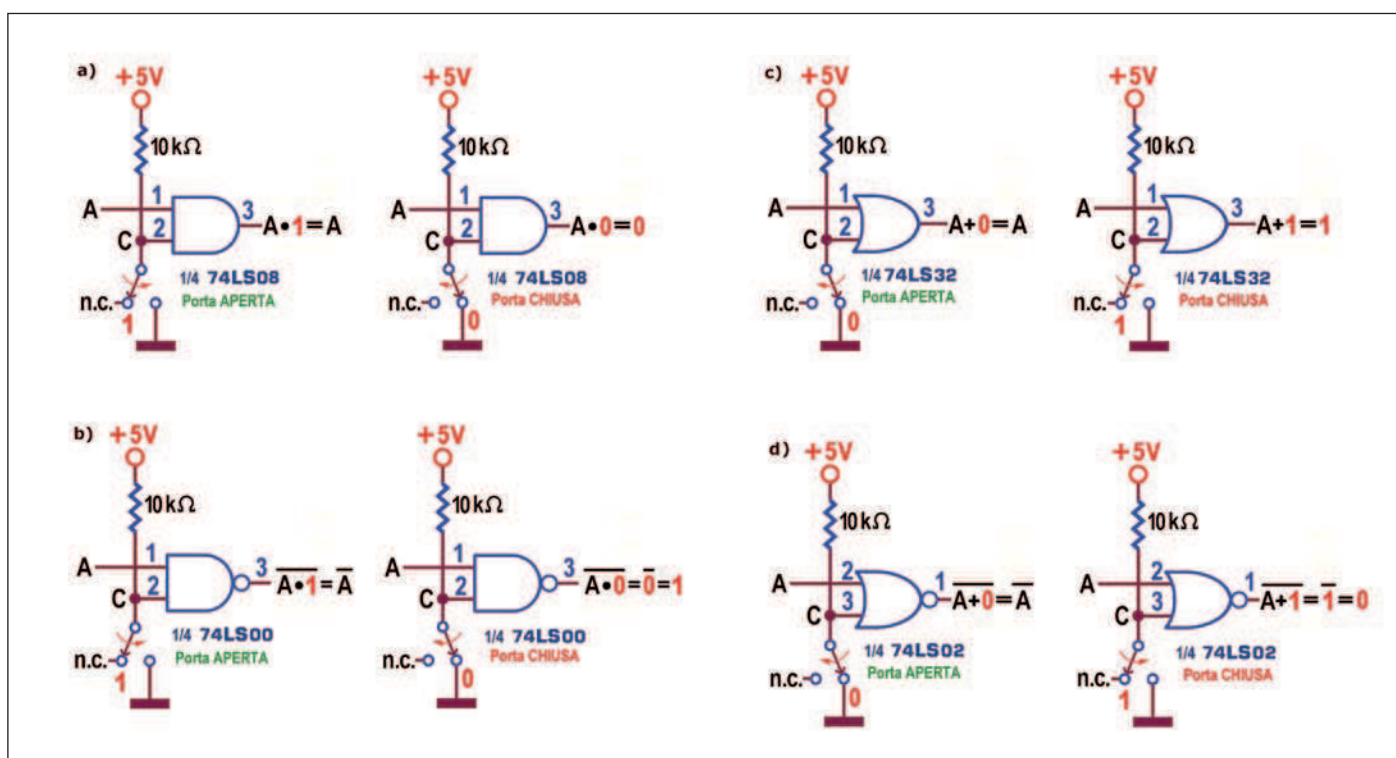



Figura 1: Gating dei segnali: controllo di un solo ingresso.

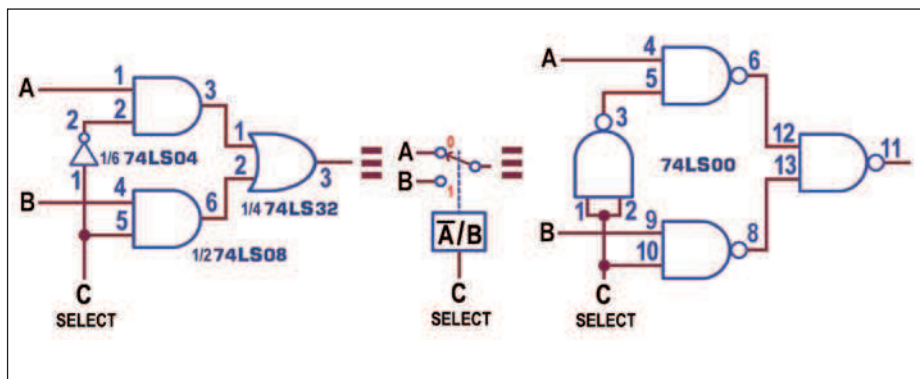



Figura 2: Gating dei segnali: controllo di due segnali d'ingresso.

Figura 3: Dual AND-OR Gates74LS58: Pin-out.

Figura 4: AND-OR-INVERTER Gates74LS54: Pin-out.

Figura 5: Quadruple 2-Line to 1-Line MUX 74LS157: Pin-out.

Figura 6: Dual 4-Line To 1-Line MUX 74LS153: Pin-out.

brare irriverente dissociare da esse la loro natura booleana e trattarle per quelle che sono: porte vere e proprie, da aprire e chiudere con il concorso di un segnale di controllo; la scorsa puntata abbiamo visto come rendere inoffensivi i piedini non necessari di una porta, stabilendo che, per garantirne il funzionamento, quelli di una OR/NOR devono essere posti a massa (0 logico) e quelli di una AND/NAND devono essere posti al positivo dell'alimentazione (1 logico).

Lo stesso principio è alla base del concetto di gating dei segnali (cioè il controllo del loro passaggio attraverso la porta stessa): ogni porta logica (gate) si presta al controllo del passaggio (gating) dei segnali: è sufficiente utilizzarne una (di qualunque tipo) a 2 ingressi, impegnando il primo per il segnale A da governare e l'altro per il segnale di controllo C; la porta coinvolta sarà ritenuta chiusa se il segnale non passa e aperta in caso contrario. Di fatto, con AND e NAND **(figure 1a e 1b)** per aprire la porta ci vorrà un 1, mentre con OR e NOR (figure 1c e 1d) ci vorrà uno 0; le figure mostrano l'azione di un deviatore meccanico posizionato per lasciar passare o meno l'ingresso A; da notare la presenza di un resistore di pull-up per garantire al segnale di controllo C un livello logico attendibile anche nella fase di transizione tra una posizione e l'altra. La f**igura 1** è a tutti gli effetti un interruttore digitale il cui stato (aperto o chiuso) è del tutto simile a quello dell'interruttore meccanico a 2 morsetti utilizzato negli impianti elettrici; in questo ambito è per altro diffuso anche l'utilizzo dei deviatori, dispositivi meccanici a 3 morsetti necessari per controllare un punto luce da 2 diverse posizioni. La realizzazione di un deviatore digitale è un classico del progetto logico, basato sulla tipica struttura AND-OR: la **figura 2** mostra 2 possibili soluzioni; in entrambe è facile constatare che l'uscita del circuito sarà sempre uguale ad uno solo dei due ingressi, in funzione del valore imposto su un terzo ingresso, chiamato ad esercitare questo controllo.

Volendo puntualizzare con pignoleria, l'azione di un deviatore sembra quella di orientare un ingresso verso 2 possibili uscite: nel nostro caso esso è utilizzato alla rovescia e, pur essendo lo stesso oggetto, sembra più logica la sua azione come selettore "da 2 a 1", creando i presupposti per un naturale aggancio alle descrizioni successive.

Aumentando il numero delle porte AND che concorrono verso l'unica OR è possibile selezionare più di 2 ingressi, sebbene questo comporti la necessità di più linee di selezione; l'evoluzione di questo punto di vista ha portato alla necessità di rendere disponibili i multiplexer. In aggiunta, la famiglia TTL offre numerosi dispositivi contenenti svariate disposizioni della struttura AND-OR; tra questi la **figura 3** mostra il 74LS58 (con 2 gruppi da 2 AND ciascuno, pin-out compatibile con il 74LS51 che, al posto della OR, propone una NOR) e la **figura 4** il mostra 74LS54 (con 1 gruppo da 4 AND, esso pure dotato di una NOR al posto della OR). Anche la serie CMOS 4000 prevede alcuni componenti di questo tipo [spesso

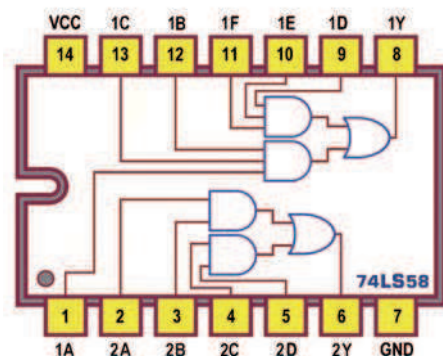
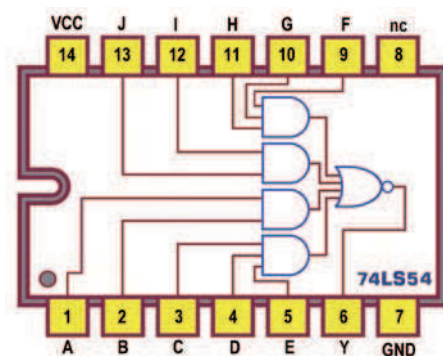
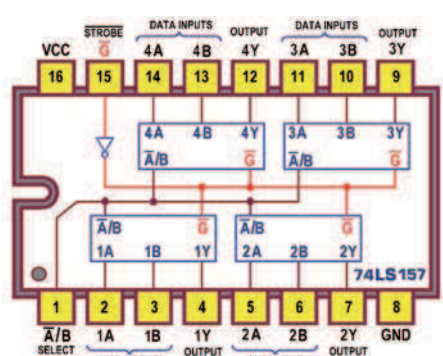
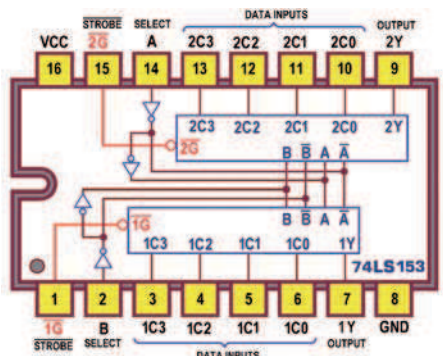
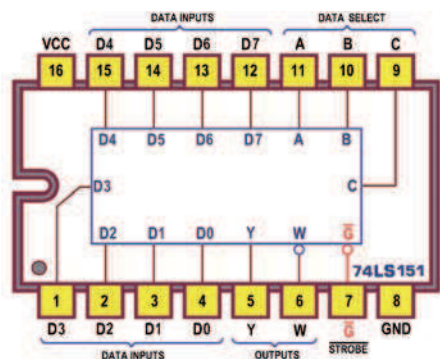



Figura 7: 8-Line To 1-Line MUX 74LS151: Pin-out.

dotati di linee di inibizione (inhibit) e/o di selezione (select) e/o di espansione (Expandable)] come il 4019 (4 gruppi, ciascuno con 2 AND a 2 ingressi in OR), il 4085 (2 gruppi, ciascuno con 2 AND a 2 ingressi in NOR), il 4086 (singolo, con 4 AND a 2 ingressi in NOR), il 4501 (componibile, contente 2 NAND a 4 ingressi e una OR/NOR), il 4506 (2 gruppi, ciascuno con 2 AND a 2 ingressi in NOR).

MULTIPLEXER (TTL):

74LS157, 74LS153, 74LS151, 74150

I Multiplexer o Selettori d'ingresso o MUX sono dispositivi chiamati a trasferire l'informazione di più ingressi (solo uno alla volta) verso l'unica uscita prevista; di norma gli ingressi sono 2, 4, 8 o 16 (una potenza di 2, in accordo con la logica binaria che caratterizza questo ambiente).

Il loro funzionamento ricorda quello di un commutatore rotativo, un componente meccanico molto utilizzato nella progettazione elettronica: rispetto ad esso questa macchina combinatoria offre il vantaggio di un controllo digitale decisamente versatile, nel senso di poter passare da un ingresso all'altro senza dover rispettare la stretta sequenza imposta dalla rotazione oraria o antioraria del perno meccanico. Di fatto la selezione dei 2^n ingressi di dato è operata dal codice binario imposto sugli n ingressi di selezione (data select), che ovviamente potrà assumere se necessario qualunque combinazione, anche non consecutiva.
La serie TTL LS prevede 4 componenti per questo dispositivo, descritto dai datasheet come Data Selector/Multiplexer: il 74LS157 contiene 4 MUX a 2 ingressi **(figura 5)**, il 74LS153 ne contiene 2 a 4 ingressi **(figura 6)**, il 74LS151 ne contiene 1 a 8 ingressi **(figura 7)** e il 74150 **(figura 8)** ne contiene 1 a 16 ingressi; scontata in quest'ultimo caso la necessità di un contenitore a 24 piedini, dato il considerevole numero di segnali coinvolti. Lo schema funzionale dei componenti combinatori (proposto da alcuni datasheet come schema logico, Logic Diagram) è decisamente interessante: sebbene piuttosto elaborato esso consente un'analisi dettagliata del loro funzionamento aiutandoci a seguire il percorso di ciascun segnale. La **figura 9** mostra i dettagli del MUX 74LS157. Appare subito evidente la struttura AND-OR di ciascuna uscita, tipica di un commutatore digitale. Ciascuno dei 2 dati di una coppia può passare in uscita (attraverso la OR) solo se il segnale che concorre con esso sulla rispettiva AND è a 1 logico (cioè la tiene aperta); e poiché ha senso che solo uno dei 2 passi in uscita, la logica di controllo del MUX deve assicurare che mentre una delle AND della coppia è aperta, l'altra deve essere chiusa. Di ciò si occupa il segnale di selezione (select, pin1): ragionando sullo schema si conferma che, con select a 0, passano gli ingressi A e con con select a 1, passano gli ingressi B (ricordiamo che il pallino, in ingresso e in uscita, si comporta come un NOT, imponendo un'inversione logica).
Molto frequentemente i dispositivi combinatori offrono la possibilità di essere "abilitati" al funzionamento: i MUX non sono da meno; nel caso in esame il segnale strobe (pin15) influenza tutte e 4 le coppie di AND, tenendole aperte solo se forzato a 0 (cioè a massa). Questo segnale attivo basso è spesso detto anche enable, evidenziando la sua fondamentale importanza per garantire funzionamento del componente; è bene precisare che, quando esso non è attivo (cioè a 1 o scollegato) il MUX "non funziona" perchè non svolge il suo compito: di fatto, qualunque sia il valore di select (che ne abiliterebbe almeno una) tutte le uscite sono infatti forzate a 0, vietando a tutti i segnali di passare e rendendo inutilizzabile il multiplexer.

Figura 9: Quadruple 2-Line to 1-Line MUX 74LS157: Schema funzionale.

74LS157

INPUT				OUTPUT
A	B	STROBE G	SELECT A/B	Y
X	X	H	X	L
L	H	L	L	L
H	X	L	L	H
X	L	L	H	L
X	H	L	H	H

H livello logico alto L livello logico basso X livello logico irrilevante

Figura 10: Quadruple 2-Line to 1-Line MUX 74LS157: Tabella di verità.

La **figura 10** presenta la Tabella di Verità (Function Table), chiamata a sintetizzare le considerazioni appena fatte: in uscita passerà l'ingresso selezionato, solo se il componente è abilitato. Lo schema pratico **(figura 11)** offre il simbolo del MUX 74LS157, da utilizzare direttamente nel contesto di un progetto, evidenziando il nome dei segnali logici che entrano e di quelli che escono e il numero dei piedini ad essi riservati.
La **figura 12** mostra una interpretazione logica del funzionamento del MUX 74LS157: esso si comporta come un commutatore "a 4 vie - 2 posizioni", cioè ogni via ha 2 ingressi e un'uscita. La serie TTL LS prevede alcuni componenti pin-out compatibili con il 74LS157: il 74LS158 è la versione con uscite negate (cioè dotata di NOR al posto della OR); il 74LS257 e il 74LS258 sono come i precedenti (uno con OR e l'altro con NOR) ma hanno uscite 3-state (in sostanza il segnale sul pin15 invece di strobe diventa output control (o output enable), forzando in alta impedenza tutte le uscite, se non è attivo, cioè a 1 o scollegato). Altri integrati (74LS298, 74LS398 e 74LS399) offrono un servizio in più, essendo in grado di mantenere il valore delle 4 uscite, in virtù della presenza, su ciascuna di esse, di una memoria edge-triggered (cioè capace di bloccarne il valore sul fronte di discesa (High to Low transition) di un segnale di clock appositamente predisposto). Meritano di essere conosciuti (sebbene fuori produzione) anche i 74LS604, 74LS606 e 74LS607, contenenti ben 8 MUX a 2 ingressi, ciascuno dotato di memoria, e con uscite negate 3-state. In concreto una linea di clock provvede a prememorizzare (sul suo fronte di salita) il valore logico delle 8 linee di entrambi i gruppi d'ingressi, A e B, su 2 registri (D-type) a 8 bit, le uscite dei quali sono spedite, a coppie, sulla tipica struttura AND-OR-NOT del MUX; il medesimo clock è utilizzato anche per controllare lo stato delle uscite, tenendole in alta impedenza quando è a 0 e abilitandole quando è a 1. Vediamo ora come è organizzato l'integrato 74LS153, contenente 2 MUX a 4 ingressi; la **figura 13** ne mostra lo schema funzionale (Logic Diagram). Osserviamo che ciascuno dei 2 MUX può ora contare su una propria linea

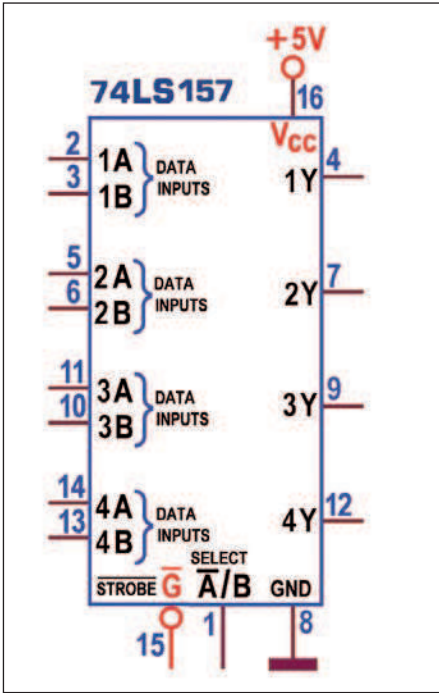



Figura 11: Quadruple 2-Line to 1-Line MUX 74LS157: Schema pratico.

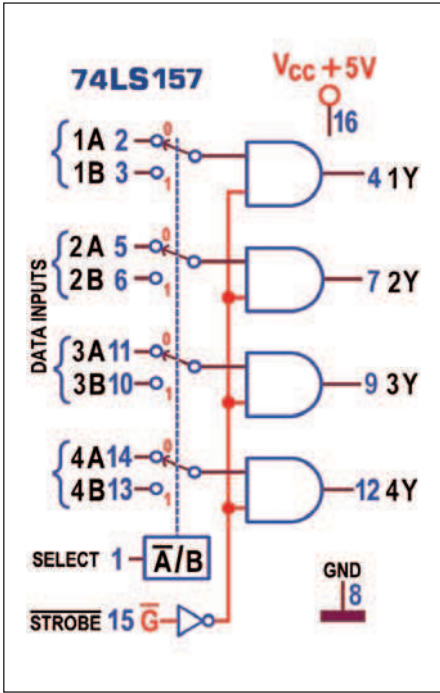



Figura 12: Quadruple 2-Line to 1-Line MUX 74LS157: Schema equivalente.

di strobe, da forzare a 0 (attiva bassa) per garantirne il funzionamento, cioè per tenere aperte tutte 4 le sue porte AND; la necessità, poi, di doverne aprire solo una alla volta impone la presenza di un codice binario a 2 bit in grado di identificarla con certezza, cioè richiede ora 2 linee di selezione.
La rete di selezione interna è dunque molto più complessa rispetto a quella del componente studiato in precedenza: seguendo i collegamenti è facile verificare che, per esempio, con select AB=00 risulterà aperta solo la AND0 di ciascun gruppo, purché anche G sia uguale a 0: solo in queste condizioni tutti e 3 i suoi ingressi di controllo sono a 1 logico, condizione indispensabile per lasciar passare il dato predisposto sul quarto, in arrivo da xC0. La **figura 14** presenta la Tabella di Verità (Function Table): l'esempio appena proposto è sintetizzato

74LS153

INPUT						OUTPUT
C0	C1	C2	C3	STROBE G	SELECT B A	Y
X	X	X	X	H	X X	L
L	X	X	X	L	L L	L
H	X	X	X	L	L L	H
X	L	X	X	L	L H	L
X	H	X	X	L	L H	H
X	X	L	X	L	H L	L
X	X	H	X	L	H L	H
X	X	X	L	L	H H	L
X	X	X	H	L	H H	H

H livello logico alto L livello logico basso X livello logico irrilevante

Figura 14: Dual 4-Line To 1-Line MUX 74LS153: Tabella di verità.

dalla seconda e dalla terza riga. Lo schema pratico da utilizzare nei nostri progetti è mostrato in **figura 15** mentre la **figura 16** offre l'interpretazione logica del suo funzionamento, cioè quello di un commutatore "a 2 vie - 4 posizioni". Sono previsti altri integrati con 2 MUX a 4 ingressi, tutti pin-out compatibili con il 74LS153: il 74LS352 è la versione con uscite negate (cioè dotata di NOR invece di OR); il 74LS253 ha uscite 3-state (la linea di strobe, ora detta output control, se a 1 o scollegata, chiude tutte le sue 4 porte AND e contemporaneamente forza in alta impedenza l'uscita della sua OR); il 74LS353 ha uscite 3-state e negate. Le 2 rimanenti versioni di MUX TTL sono entrambe riconducibili ad un commutatore "ad una via", cioè con un'unica uscita, ma con una notevole quantità di ingressi di dato (8 per il 74LS151 e 16 per il 74150) e di ingressi di selezione (select) tale da assicurare un codice binario (a 3 bit per il primo e a 4 bit per il secondo) adeguato a garantire le 2^n combinazioni necessarie. Il loro schema funzionale è sempre lo stesso, basato sulla struttura AND-OR, ma diventa più ingombrante per maggiore quantità di ingressi richiesta da ciascuna porta logica coinvolta; la **figura 17** mostra una interpretazione logica del MUX a 8 ingressi, lasciando al lettore la capacità di immaginare quella del MUX a 16 ingressi. Le **figure 18** e **19** suggeriscono invece per entrambi lo schema pratico, da utilizzare nei progetti, consigliato per la sua sintetica completezza: possiamo sottolineare la particolarità del 74LS151 di offrire l'uscita sia in forma diretta che in forma negata. Del 74LS151 esiste anche la versione pin-out compatibile 74LS251 con uscite 3-state (la cui linea output control funziona nel modo visto negli analoghi casi precedenti) e la serie (fuori produzione) dei 74LS354, 74LS355 e 74LS356, dotati di memoria sia sugli ingressi di dato che su quelli di selezione e con uscite 3-state. La sigla MUX visibile nei simboli logici di figura 20 identifica i multiplexer, nello standard IEEE (stabilito dalle norme internazionali); per essi è prevista una notazione speciale che identifica 2 zone ben distinte:

- la metà superiore organizza gli ingressi di selezione: se il MUX prevede più di una via (come il 74LS157 e il 74LS153) la grafica include un blocco di controllo comune, appoggiato su di esse, per sottolineare che gli ingressi di select influenzano tutte le sottostanti sezioni; se sono più di uno il primo è identificato da 0 e l'ultimo da n, il numero che indica il rispettivo peso, cioè l'esponente della potenza di 2 che stabilisce quale dei possibili ingressi sarà abilitato a passare in uscita (se il segnale G è attivo); il tutto viene ribadito dalla parentesi graffa che li raccoglie e li fa corrispondere proprio a G, alla destra del quale la grafica $0/(2^n-1)$ non è una frazione ma indica l'intervallo previsto per gli ingressi di dato, per esempio da 0 a 3 (74LS153) o da 0 a 7 (74LS151);
- la notazione con la graffa non è necessaria per il 74LS157 perché ha un solo ingresso di selezione, in questo caso indicato con G1 per sottolineare che influenzerà i segnali di dato recanti (nella metà inferiore) l'identificatore "1" (se G1 vale 1) o "1 negato", se G1 vale 0);
- la metà inferiore propone invece tutti gli ingressi di dato separati (se necessario) in sezioni, all'interno delle quali sono numerati da 0 a 2^n-1; se ne è prevista più di una (4 per 74LS157, 2 per 74LS153) le etichette sono presenti solo nella prima;
- se il segnale di abilitazione (indicato con EN) è unico per tutte le (eventuali) sezioni viene incluso nella metà superiore; altrimenti ciascuna sezione avrà il proprio segnale G.

Le caratteristiche elettriche dei dispositivi multiplexer TTL LS sono quelle della famiglia, tra le quali ricordiamo la I_{OL}=8 mA (corrente massima assorbita in uscita a livello 0) e la V_{OH}=3,5 V (tensione tipica in uscita a livello 1). La potenza dissipata massima (con uscite aperte e ingressi a 0) va da 80 mW (74LS157) a 350 mW

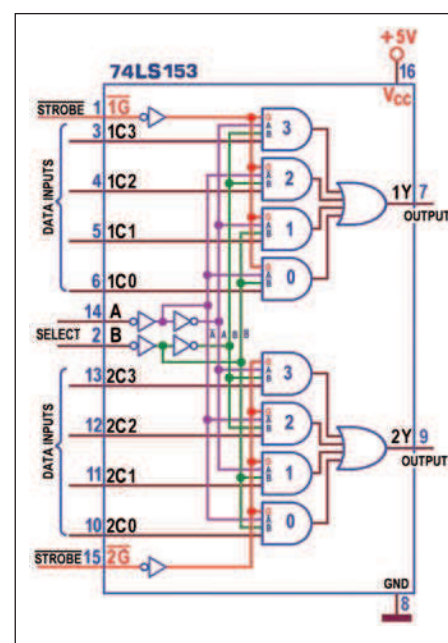



Figura 13: Dual 4-Line To 1-Line MUX 74LS153: Schema funzionale.

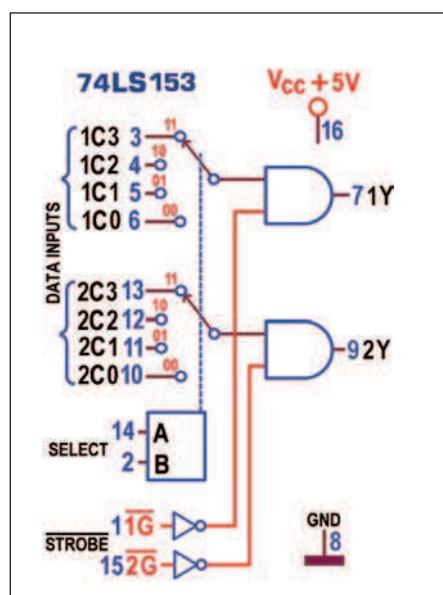



Figura 16: Dual 4-Line To 1-Line MUX 74LS153: Schema equivalente.

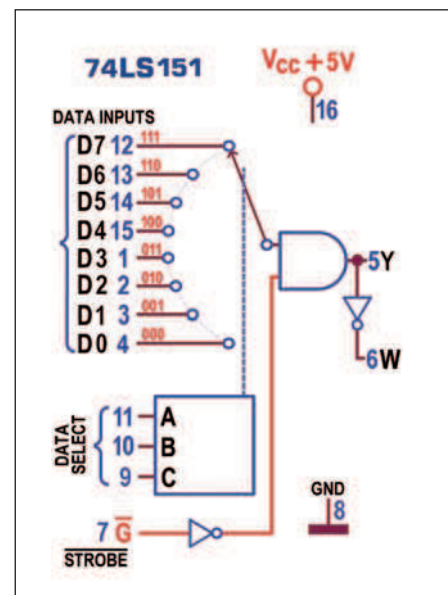



Figura 17: 8-Line To 1-Line MUX 74LS151: Schema equivalente.

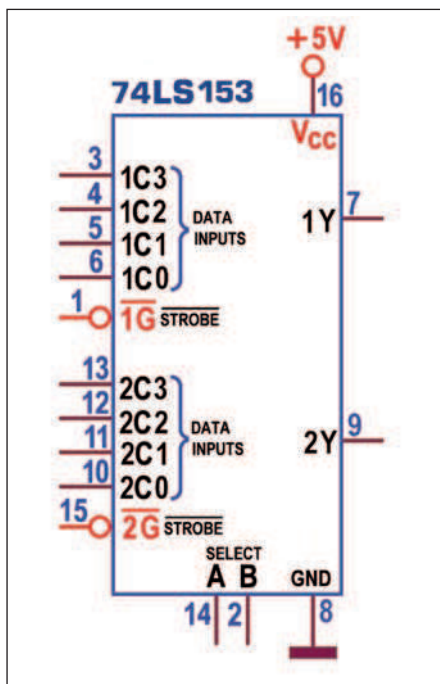



Figura 15: Dual 4-Line To 1-Line MUX 74LS153: Schema pratico.

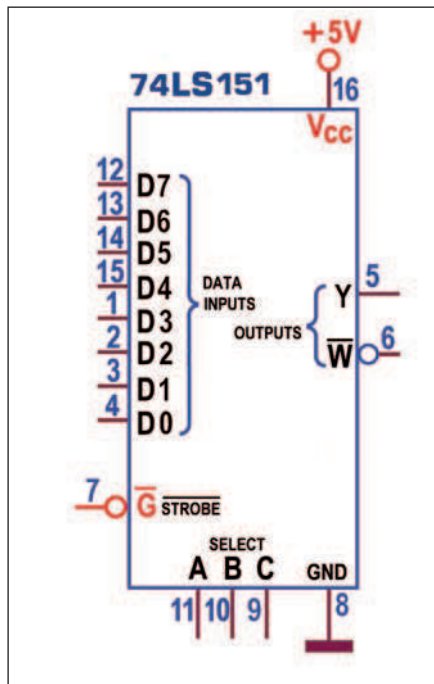



Figura 18: 8-Line To 1-Line MUX 74LS151: Schema pratico.

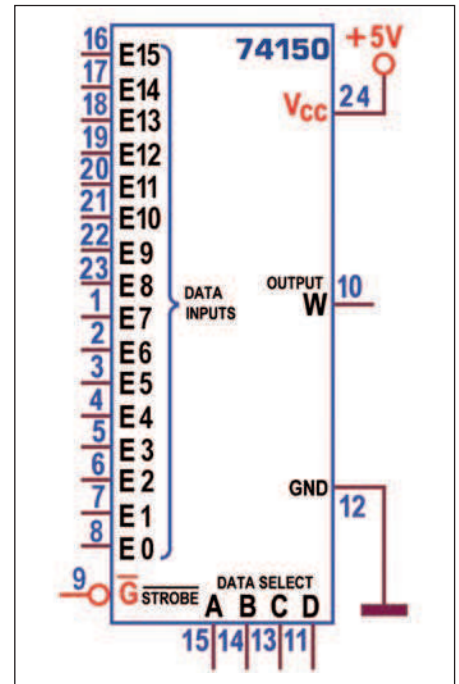



Figura 19: 16-Line To 1-Line MUX 74150: Schema pratico.

Figura 20: MUX [TTL]: Simbolo logico ANSI/IEEE Std. 91-1984.

Figura 21: 1/2 4-Line To 1-Line MUX 74LS153: generatore di funzioni.

Figura 22: Quadruple 2-Line to 1-Line MUX 4519: Pin-out.

(74LS151) mentre il ritardo di propagazione massimo (Propagation Delay Time, con carico di 2kohm/15pF) è mediamente di 27 ns per le transizioni da ingresso dato a uscita, e leggermente maggiore da strobe o da select a uscita; il 74150 (TTL standard) oltre a disporre di correnti massime più elevate (I_{OL}=16 mA, assorbita in uscita) dissipa al massimo 340 mW ed oppone un ritardo massimo da 35 ns (t_{PLH}) a 55 ns (t_{PHL}) con carico di 400ohm/15pF. In chiusura vogliamo ricordare come può essere utilizzato il multiplexer: la possibilità di selezionare una delle sue 2^n entrate è intrinseca della sua natura di selettore, fissando a piacere un qualunque codice binario a n bit sulle n linee di select; organizzando su questi ultimi una sequenza continua (per esempio da 000 a 111, su un 74LS151) è facile pensare ad semplice un convertitore parallelo-serie. Ma l'utilizzo più curioso è quello di generatore di funzioni logiche: disponendo della Tabella di verità della funzione booleana da realizzare basta fissare sui 2^n ingressi di dato (con degli interruttori, switch) il livello logico di volta in volta da essa previsto in uscita, in corrispondenza delle rispettive 2^n combinazioni; ciò fatto, gli n ingressi di selezione e l'uscita del MUX si possono ritenere anche ingressi ed uscita del circuito logico simulato. La **figura 21** chiarisce il concetto con 1/2 74LS153 chiamato a simulare una OREX.

MULTIPLEXER (CMOS):

4519, 4539, 4512

La serie CMOS 4000 prevede il 40257 e il 4519 **(figura 22)** con 4 MUX a 2 ingressi, il 4539 con 2 MUX a 4 ingressi e il 4512 con un solo MUX a 8 ingressi (in questo caso con uscita 3-state). Questa serie prevede anche numerosi multiplexers analogici (che funzionano perfettamente anche in digitale) dotati di interruttori (switch) bidirezionali controllati digitalmente, come il 4053 (triplo MUX a 2 ingressi), il 4052 (doppio a 4 ingressi), il 4051 (singolo a 8 ingressi), il 4529 (doppio a 4 ingressi o singolo a 8 ingressi), il 4067 (singolo a 16 ingressi) e il 4097 (doppio a 8 ingressi).

Ciascuno degli interruttori bidirezionali analogici presenti (2, 4, 8 o 16) è collegato da una parte ad una linea indipendente e dall'altra ad una linea in comune con gli altri switch; ogni linea è intrinsecamente di input/output ed è quindi possibile scambiarne i ruoli, facendo assumere agli ingressi la funzione di uscite e all'uscita quella d'ingresso. In conclusione questi MUX possono essere usati anche come demultiplexer DMUX, che vedremo la prossima puntata. ❏

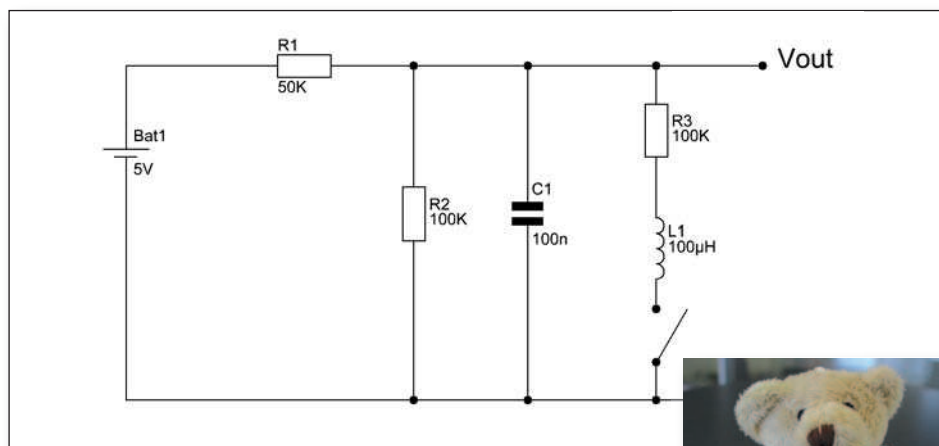
Codice MIP 2756763

elettro quiz

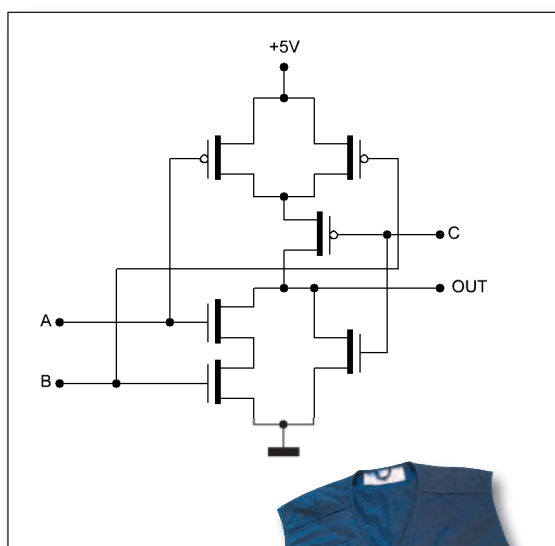
288

Le risposte ai quiz “Base” e “Avanzato” vanno inviate esclusivamente compilando il modulo su www.farelettronica.com/eq specificando la parola chiave ***“millmann”****.*

Le risposte ed i vincitori (previa autorizzazione) sono pubblicati alla pagina www.farelettronica.com/eq a partire dal 15 del mese successivo alla pubblicazione sulla rivista.

A tutti i partecipanti verrà assegnato un buono sconto del 10% (validità 3 mesi dalla data di assegnazione) utilizzabile per un prossimo acquisto su www.ieshop.it

base

Dato il circuito di figura, determinare il valore della tensione di uscita dopo 120 secondi dalla chiusura dell’interruttore.

Se rispondi correttamente potrai vincere il simpatico portachiavi di Fare Elettronica.

avanzato

Determinare la funzione logica svolta dal circuito di figura.

Per i più bravi in palio il gilet “Reporter” di Fare Elettronica.

Interfaccia con I CIRCUITI *integrati*

Una volta progettato un circuito digitale, è importante sapere come interfacciare le sue uscite ai carichi che normalmente richiedono tensioni e correnti ben più elevate di quelle fornite dai circuiti integrati digitali

In questa trattazione ci occuperemo dell'interfacciamento dei circuiti integrati più utilizzati: CMOS, TTL e compatibili, anche se quanto di seguito esposto consentirà di interfacciare l'uscita di qualsiasi componente elettronico. Prima di addentrarci nella trattazione è necessario soffermarsi sulle caratteristiche fondamentali dei componenti scelti per realizzare l'interfacciamento. Ovviamente tratteremo i più comuni e precisamente:

a) Transistori
b) Diodi zener
c) Triac
d) Relè
e) Fotoaccoppiatori

TRANSISTOR

E' il componente più utilizzato ed è affidabile ed economico. Il funzionamento che si richiede al transistor nell'interfacciamento è quello di trasformare una uscita dell'integrato, caratterizzata da un certo valore di corrente e di tensione, in una uscita con valore di tensione e corrente diversi. Si esamini il circuito di **figura**

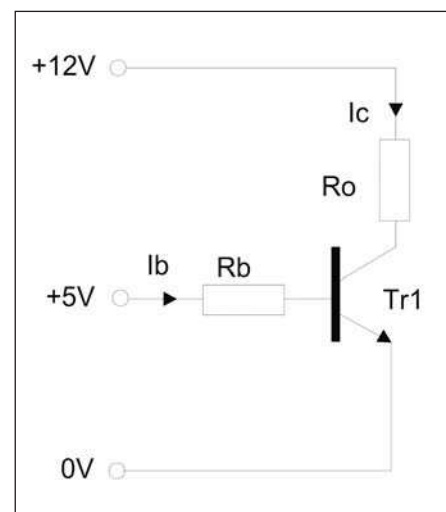



Figura 1: il transistor all'uscita di una porta logica.

1. Il problema che ci poniamo è il seguente: abbiamo l'uscita Vi proveniente da un integrato TTL con una tensione di 5V e una corrente massima di 10mA.
Tale uscita non può alimentare la resistenza di carico Ro che necessita di una tensione di 12V e, avendo un valore di resistenza di 39 Ohm, di una corrente Ic=12V/(39 Ohm)=0,3 Ampere. Il dato base occorrente perché il transistore soddisfi la condizione posta è:

$$h_{FE} = Ic/Ib = 0,3A/0,01A > 30$$

Questa condizione ci assicura che il transistore funzioni in commutazione. Dunque, trascurando la tensione base emettitore (di pochi decimi di Volt), inserendo fra base ed emettitore una resistenza pari a

$$Rb = 5V/10mA = 500\ Ohm$$

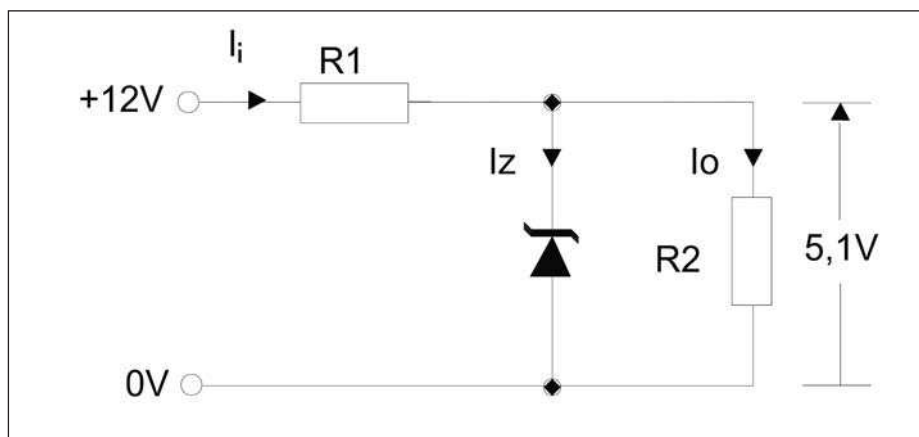



Figura 2: livellamento dell'uscita con diodo zener.

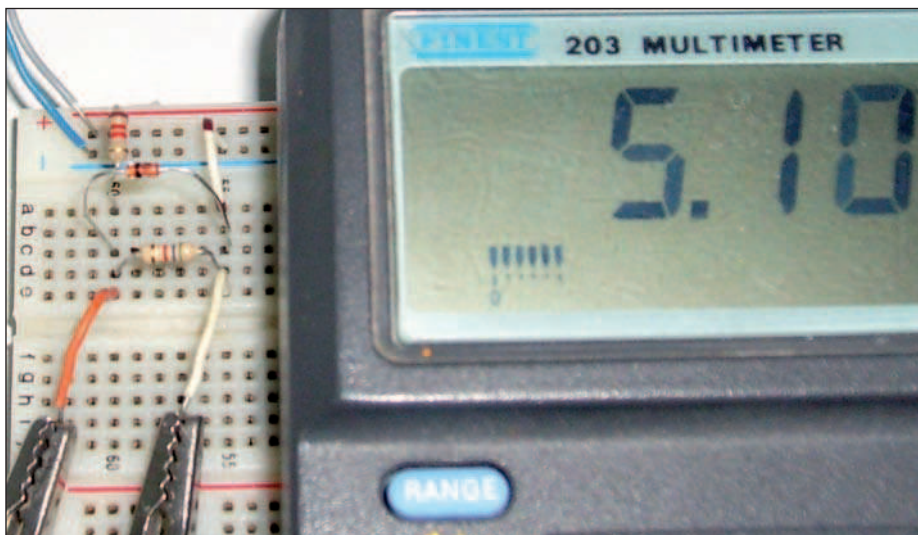



Figura 3: misura della tensione di uscita al circuito di figura 2.

(valore commerciale di 560 Ohm), il transistore sarà in grado di erogare una corrente di 0,3 A. In realtà è opportuno che il valore di h_{FE} (guadagno di corrente in corrente continua) sia sensibilmente superiore al valore calcolato per essere sicuri che il transistore funzioni in regime di saturazione. Adesso siamo in grado di scegliere il transistore occorrente allo scopo. Esso deve soddisfare le seguenti caratteristiche:

1) V_{CE0}: tensione in c.c. tra collettore ed emettitore > 12V

2) V_{BE0}: tensione in c.c. tra base ed emettitore > 5V

3) h_{FE}: guadagno in c.c. di corrente > 30

4) I_C: corrente di collettore in c.c. > 0,3 Ampere

Figura 4: uso del triac come interfaccia di uscita.

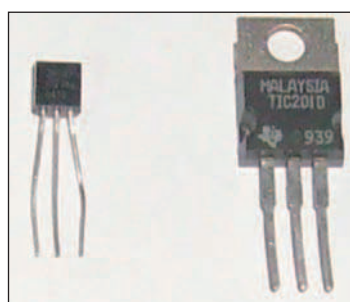



Figura 5: il triac Z0107 (a sinistra) e il triac TIC201D (a destra).

A questo punto occorre esaminare i datasheet dei transistori (i dati tecnici forniti dal costruttore) e scegliere Tr1. Per esempio il transistor NTE123AP soddisfa i requisiti richiesti in quanto i suoi parametri caratteristici sono i seguenti:

- V_{CE0}=40V > 12V
- V_{BE0}=6V > 5V
- h_{FE}=80 > 30 (si noti che tale valore è dato per V_{CE}=1 Volt, cioè quando il transistor è prossimo alla saturazione, poiché in tal caso V_{CEsat}=0,4 Volt)
- I_C=600mA > 300mA

E' comunque importante scegliere un componente che soddisfi le condizioni poste entro certi limiti. Precisamente non è opportuna, sia dal punto di vista economico che dal punto di vista tecnico, la scelta di un componente eccessivamente sovradimensionato.

Figura 6: l'impiego di un relè per il pilotaggio di un carico alla tensione di rete.

DIODO ZENER

La caratteristica fondamentale del diodo zener è quella che, polarizzato inversamente ed entro certi limiti, mantiene una tensione ai suoi capi pressoché costante. I limiti, in partenza, sono determinati dalla potenza massima fissata dal datasheet del diodo zener. Tipicamente le potenze più utilizzate sono 0,5Watt e 1

Figura 7: tre tipi di relè comunemente adoperati nelle schede elettroniche.

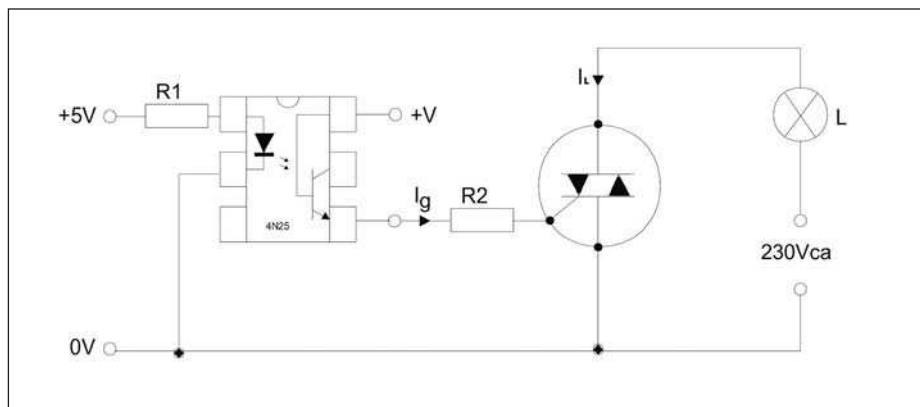



Figura 8: l'impiego di un foto accoppiatore.

Watt. Esaminiamo il circuito di **figura 2**. Poniamo di avere una uscita da un integrato 12V e una corrente che non superi 10mA e si voglia interfacciare con l'ingresso di un integrato TTL che necessita della tensione di 5 Volt e di una corrente di 1mA circa. Allora sarà:

R2=5,1V/1mA=5k
che arrotondiamo a 5,6k

Scegliamo un diodo zener che più si avvicini a 5 Volt. Il valore commerciale disponibile più vicino è di 5,1 Volt. Esaminiamo il datasheet del zener BZX79C5V1: la corrente di zener è 5mA. Ciò vuol dire che, se la corrente di zener è 5mA, allora:

- Io=Vz/R2=5,1V/5,6k=0,9mA
- VR1=Vi-Vz=12V-5,1V=6,9V
- Ii=Iz+Io=5mA+0,9mA=5,9mA
- R1=VR1/Ii=6,9V/5,9mA=1,17k che arrotondiamo a 1,2k

In **figura 3** è mostrato il circuito sperimentale in cui viene misurata la tensione ai capi del diodo mediante un multimetro digitale.

TRIAC

Il triac, così come il relè, è molto utile quando si deve pilotare un carico a 230V in alternata disponendo di una tensione

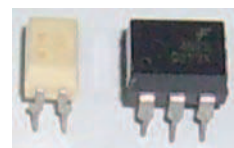



Figura 9: due tipi di accoppiatori: il P626 in c.a. (a sinistra) e il 4N25 in c.c. (a destra).

continua. A tale scopo esaminiamo il circuito di **figura 4**. Poniamo che Vi sia l'uscita di un integrato TTL (5V).
Si voglia pilotare l'accensione o lo spegnimento di una lampada che abbia la potenza di 100Watt. Per un integrato TTL la corrente massima disponibile per ciascuna uscita è di 10mA. Si deve, allora, trovare un triac che abbia le seguenti caratteristiche:

- Ig < 10mA
- Vak > 230V
- IL=100W/230V=0,43Ampere

Dove:

- Ig è la corrente che porta in conduzione il triac
- Vgo è la tensione fra il gate e massa
- Vak è la tensione fra anodo e catodo
- Il è la corrente che circola sulla lampada L

Analizziamo il dasheet del triac TIC201D si leggono i valori:

- Igt=Ig=5mA (per valori positivi) < 10mA
- Vak=400V>230V
- IT(rms)=Il=2,5Ampere > 0,43A

Quindi questo triac, che è il più piccolo della serie TIC, soddisfa ampiamente i dati di partenza. Lo stesso risultato si sarebbe ottenuto, con risparmio sui costi e sullo spazio occupato utilizzando il triac Z0107, infatti tutte le condizioni sopra esposte sono verificate. In **figura 5** sono visibili a sinistra il triac Z0107 e a destra il triac TIC201D.

RELÈ

Come abbiamo visto prima, il triac ci consente di alimentare un carico in corrente alternata. Però analizzando il circuito di **figura 4** si riscontra immediatamente un dato importante: la linea di alimentazione del circuito integrato, che è 5 Volt in c.c., ha in comune la fase o il neutro del circuito alimentato in corrente alternata. Quando ciò, per determinate applicazioni, non è accettabile allora si ricorre al relè che dispone di contatti liberi. Cerchiamo di risolvere tale problema, tenendo fermi i dati precedenti, con il circuito di **figura 6**.

La prima domanda che ci si può porre è la seguente: perché occorre il transistor TR1? La risposta è che il relè comunemente impiegato nei circuiti elettronici per eccitarsi ha bisogno di essere attraversato nella sua bobina da una corrente che va tra i 30mA e i 50mA. Assumiamo nel nostro caso il valore di 30mA. Poiché il circuito d'ingresso può fornire una corrente massima di 10mA, se ne deduce che non può pilotare direttamente il relè. La scelta del transistore ricadrà su un transistore di piccola potenza. Per esempio il comunissimo ed economicissimo BC547C. Esso ha un h_{FE}=Ic/Ib=300 . Poiché Ic=0,03Ampere ne consegue che perché si abbia la saturazione ed il funzionamento in commutazione del transistore che Ib=0,03Ampere/300=0,1mA. Da cui si deduce che una corrente Ib=1mA porta certamente il transistor in commutazione. Di conseguenza la resistenza Ri sarà:

Ri=5V/0,001Ampere=5k
che viene arrotondata a 4,7k

La presenza del diodo D1 evita che quando il relè passa dalla conduzione all'interdizione la tensione tra collettore ed emettitore divenga doppia. In **figura 7** sono raffigurati i quattro relè comunemente adoperati nelle schede elettroniche.

FOTOACCOPPIATORI

Il loro funzionamento si basa sul fatto che il fotodiodo, opportunamente alimentato, attiva il fototransistore. Anch'essi sono adoperati quando si rende necessario separare il circuito di comando dal circuito utilizzatore ed in sostituzione, quando ciò è possibile, del relè. Prendiamo in esame, ancora una volta, il problema dell'alimentazione di una lampada da 100Watt da pilotare con un integrato TTL. Esaminiamo la **figura 8**. Da essa si evince che riusciamo a separare il circuito che alimenta l'integrato TTL dalla rete a 230V, Però nasce l'onere di alimentare il fototransistore con una tensione continua. Il problema non si pone se il carico necessita di una alimentazione in c.c. Vi è da aggiungere che vi sono fotoaccoppiatori alimentati in c.a., che alcuni hanno come uscita circuiti logici etc. In **figura 9** sono riportati due tipi di accoppiatori: il P626 in c.a. (a sinistra) e il 4N25 in c.c. a destra. Nel prossimo numero esamineremo circuiti più complessi che investiranno i componenti sopra trattati e i circuiti integrati TTL, CMOS e similari. ❑

CODICE MIP 2756789

di FRANCESCO PENTELLA

Lezioni di VHDL *(parte terza)*

Sintassi e semantica VHDL

In questa terza parte continuiamo a soffermarci sugli aspetti lessicali e semantici del linguaggio: componenti essenziali per scrivere qualsiasi programma in VHDL

I signal, come abbiamo visto la scorsa puntata, sono oggetti che connettono direttamente elementi concorrenti. Ogni oggetto, prima di essere utilizzato, deve essere prima dichiarato associandogli un tipo. Un signal può essere locale o globale. La dichiarazione di un signal all'interno dell'architettura (architecture) definisce l'elemento locale, mentre all'interno di un package lo rende globale. I signal, ricordiamo, sono utilizzati per trasportare le informazioni tra elementi funzionali.

LETTERALI

Agli oggetti VHDL (variabili, costanti o signal) possono essere assegnati valori o espressioni, questi sono anche definiti letterali. I letterali possono essere utilizzati in differenti modi e dipendono sempre dal contesto. In VHDL i letterali di tipo valore carattere sono composti racchiudendo il loro equivalente carattere, in base al codice ASCII relativo, tra apici o due apostrofi (single quotation mark): 'a', 'B', ','. In base a questo sistema è pos-

TABELLA 1 – OPERATORI E ORDINE DI PRECEDENZA

Operatori – La prima colonna evidenzia la lista dei operatori in ordine di precedenza, si può notare che i operatori logici hanno la precedenza più bassa.

CLASSE							
LOGICI Accettano due operandi (eccetto not) di tipo bit, boolean e bit_vector. Restituiscono un valore dello stesso tipo.	And	Or	Nand	Nor	Xor	Xnor	Not
RELAZIONALI Accettano due operandi dello stesso tipo bit, boolean, bit_vector, integer, real, character, string o time. Restituiscono un valore di tipo boolean.	=	/=	<	<=	>	>=	
ROTAZIONE E TRASLAZIONE Accettano due operandi. Il primo è un array a una dimensione di bit o boolean, il secondo è il numero di posizioni a spostare. Il risultato dell'operazione ricade nel tipo di sinistra.	Sll	Srl	Sla	Sra	Rol	Ror	
ADDIZIONE Accettano due operandi (eccetto abs) dello stesso tipo integer, real, character o time.	+	-	&				
UNARI	+ (segno)	- (segno)					
MOLTIPLICAZIONE Accettano due operandi (eccetto abs) dello stesso tipo integer, real, character o time.	*	/	Mod	Rem			
ALTRI Accettano due operandi (eccetto abs) dello stesso tipo integer, real, character o time. Non si applica all'operazione di concatenazione.	**	Abs	&				

sibile rappresentare tutti i caratteri ASCII stampabili, così con ‘A’ si identifica la lettera maiuscola A, mentre con ‘z’, si la lettera minuscola, con ‘,’ è possibile rappresentare il carattere virgola, così ‘’‘ i doppi apici o con ‘ ‘ il carattere spazio. Una nota sul tipo std_logic. Sebbene gli identificatori non sono case-sensitive, l’interpretazione dei valori letterali è, al contrario, case-sensitive. Questo significa che ‘z’ non è equivalente a ‘Z’. Non solo, una stringa è una sequenza di caratteri racchiusi tra doppi apici o tra i caratteri %, così in questo modo “Una stringa” è l’equivalente a %Una stringa&.. Con le stringhe diventa possibile l’operazione di concatenazione, espressa con l’operatore &. L’operazione di concatenazione permette di legare diverse stringhe divise in linee separate. Esistono poi le stringhe di bit, o bitstring. Una bit-string è una sequenza di valori che rappresentano bit (binary digits) che possono assumere un valore di ‘0’ o ‘1’. Una rappresentazione del genere è costituita da una stringa racchiusa tra doppi apici (double quotation mark), o utilizzando il carattere %, preceduti da un carattere che ne specifichi la base. Questa può assumere, indifferentemente, il carattere B per binario, O per ottale (base otto) o X per esadecimale (base 16). Con B”0100011” di rappresenta la forma binaria del valore decimale 35, mentre con B” 010_0011” otteniamo un valore equivalente; si nota che si è utilizzato il carattere di sottolineatura per aumentare la leggibilità. La base, poi, può essere scritta indifferentemente in maiuscolo o in minuscolo. Esistono poi i letterali fisici che sono utilizzati per rappresentare una misura di una grandezza fisica espressa come un numero intero seguito da un’unità di misura. La sintassi che occorre utilizzare è così evidenziata:

```
bit_string_literal ::= base_specifier “ bit_value “
base_specifier ::= B | O | X
bit_value ::= extended_digit { [ underline ] extended_digit }
```

TIPI DI DATI

In ogni linguaggio di programmazione, lo scopo di un programma è quello di manipolare diversi oggetti. Ogni oggetto ha sua specifica proprietà che dipende dal tipo che questo assume; a questo scopo, l’oggetto può assumere determinati valori e può essere utilizzato con differenti operazioni. Il tipo, di conseguenza, di un oggetto permette di identificare le seguenti informazioni: i valori che possono assumere gli oggetti appartenenti al tipo e le operazioni che si possono compiere con gli oggetti appartenenti al tipo stesso. È nella fase dichiarativa che si associa un oggetto ad un determinato tipo. Il VHDL è un linguaggio fortemente tipizzato, questo vuol dire che ogni oggetto prima di essere utilizzato deve essere dichiarato. In generale, poi, non è consentito assegnare un valore ad un oggetto di un altro tipo senza una esplicita conversione. In VHDL ci sono quattro tipi di dati: scalari (costituiti da un solo valore), composti (insieme di valori), access (puntatori) e file types. Il tipo scalare rappresenta un singolo valore ed è ordinato, in questo modo è possibile utilizzare operazioni relazionali. Viceversa, il tipo composto è costituito da più valori di tipo scalare: se sono dello stesso tipo è uno array, mentre con tipi differenti si rientra nella tipologia di un record. Il linguaggio VHDL prevede, poi, alcuni tipi predefiniti. Le **tabelle 1** e **2** mostrano l’ordine di precedenza e gli operatori utilizzati. Il tipo bit è il più semplice tipo disponibile in VHDL. Tale tipo rappresenta un valore binario e può unicamente assumere i valori logici ‘0’ ed ‘1’. Si noti che le costanti 0 ed uno devono essere racchiuse tra singoli apici (quindi ’0’ e ’1’) per distinguerle dai valori numerici interi 0 ed 1. Gli operatori definiti per tale tipo sono soltanto gli operatori di assegnamento, gli operatori confronto e gli operatori logici. Il listato seguente mostra una possibile applicazione:

```
architecture esempio of esempio1 is
signal es1 : bit;
   begin
      neq <= not es1; — neq attivo basso
      es1 <= ‘1’ when (a = b) else ‘0’;
      eq <= es1;
   end esempio;
```

Al contrario, a volte può essere conveniente raggruppare più segnali utilizzando una singola rappresentazione. In questo modo diventa utile disporre di un array. A questo scopo si ricorre al tipo bit_vector; infatti, con il tipo bit_vector si vuole rappresentare un array di bit. Un bit vector è sostanzialmente un insieme di segnali contraddistinti da un nome e da un indice. È possibile accedere in lettura o in scrittura ai vari elementi del vettore mediante l’uso di indici. Con la definizione:

```
b: in bit_vector (7 downto 0)
```

si definisce un vettore di 8 bit. In questo caso la parola chiave ‘downto’ (ordinamento discendete) definisce l’ordinamento dei bit che compongono il vettore. In VHDL un ordinamento di tipo ascendete, invece, viene espresso attraverso la parola chiave ‘to’.

Occorre notare che i tipi bit e bit_vector non sono comunemente utilizzati in VHDL, in quanto non consentono, ad esempio, di definire che un segnale sia in condizioni di alta impedenza.

Un altro esempio di tipo predefinito è il tipo booleano. In sostanza questo è uno scalare enumerativo che può assumere due valori: true o false.

Si dispone poi del tipo character che è uno scalare enumerativo con valore carattere. Il VHDL consente, inoltre, di definire dei vettori in cui gli indici non sono indicati (array uncostrained), ma sono dichiarati successivamente.

Un esempio è il seguente, tratto dalla libreria ieee 1164:

```
type std_logic_vector is array (natural range <>) of std_logic;
```

si utilizza il tipo predefinito natural che comprende tutti gli interi maggiori o uguali a zero. Array di dimensioni non specificate sono utilizzati, ad esempio, per definire funzioni e procedure in grado di operare con parametri di dimensioni generiche.

Il tipo integer ed i corrispondenti operatori relazionali ed aritmetici sono predefiniti in VHDL; esso include tutti i numeri interi compresi fra $-(2^{31}-1)$ e $2^{31}-1$. In una descrizione VHDL orientata alla sintesi, i segnali e le variabili di tipo integer dovrebbe sempre essere costretti all’interno

TABELLA 2 – OPERATORI RELAZIONALI			
OPERATOR	DESCRIPTION	OPERAND TYPES	RESULT TYPE
=	Equality	any type	Boolean
/=	Inequality	any type	Boolean
<	Smaller than	scalar or discrete array types	Boolean
<=	Smaller than or equal	scalar or discrete array types	Boolean
>	Greater than	scalar or discrete array types	Boolean
>=	Greater than or equal	scalar or discrete array types	Boolean

TABELLA 3 – ALCUNE FUNZIONI DI CONVERSIONE SUPPORTATE DAL PACKAGE STD_LOGIC_1164

FUNZIONE	DA	A
To_Bit	Std_logic e std_ulogic	Bit
To_StdULogic	Bit	Std_ulogic
To_StdUlogicVector	Std_logic e bit_vector	Std_ulogic_vector
To_BitVector	Std_logic_vector e std_ulogic_vector	Std_ulogic_vector
To_StdLogicVector	std_ulogic_vector	std_logic_vector
To_StdLogicVector	bit_vector	std_logic_vector

di un range predefinito. Il VHDL ha due sottotipi interi predefiniti:

```
subtype natural is integer range 0 to
higest_integer;
subtype positive is integer range 1
to higest_integer;
```

dove higest_integer è pari a: $2^{31}-1$ = =2.147483647.

ARRAY

Un array nel linguaggio VHDL è una collezione di elementi indicizzabili tutte dello stesso tipo. Un array può essere monodimnsionale (cioè con un indice) o multidimensionale (con un numero variabile di indici). Inoltre, un array può essere constrained, cioè i limiti dell'array sono definiti al momento della sua dichiarazione, o unconstrained, in questo caso i suoi limiti sono stabiliti successivamente. Un array può essere dichiarato come:

```
constant dimensione : integer := 9;
type esempio is array (dimensione-1
downto 0) of std_logic;
type array_multi is array (1 to 8, 1
to 8) of real;
type banco_registri is array (byte
range 0 to 256) of integer;
```

Per accedere agli elementi di un segnale o di una variabile di tipo array è sufficiente utilizzare il nome del segnale, o della variabile, con gli indici indicati fra parentesi. Ad esempio, con l'array:

```
type a is array (1 to 4) of character;
```

si accede ai singoli elementi dell'array attraverso una notazione posizionale; infatti con a(1) si accede al primo elemento, mentre con a(n) all'elemento ennesimo. È possibile utilizzare poi la parola chiave others per inizializzare un array ad un valore prefissato. In questo caso, per inizializzare un array o possiamo scrivere, utilizzando un ciclo for, tutti gli elementi incrementando gli indici oppure ricorrere a others, in questo modo:

('1', others => '0')

Un aspetto da tenere presente è l'effetto sull'ordine di assegnamento per la direzione degli array; infatti, gli elementi sono ordinati per posizione da sinistra a destra e non secondo gli indici. Se il range è crescente, l'indice dell'elemento più significativo a sinistra è il più basso. Se il range è decrescente, l'indice dell'elemento più significativo a sinistra è il più alto. Ad esempio, se il segnale *X* è dichiarato e inizializzato così:

```
signal X : bit_vector (0 to 3) :=
('1','1','0','0');
```

gli elementi di X ordinate da sinistra a destra sono, rispettivamente, X(0)=1, X(1)=1, X(2)= 0, X(3)=0. Invece, con:

```
signal X : bit_vector (3 to 0) :=
('1','1','0','0');
```

gli elementi di X ordinate da sinistra a destra sono ora X(3)=1, X(2)=1, X(1)= 0, X(0)=0.

RECORD

Un array ha anche diverse limitazioni, per esempio tutte le componenti devono essere dello stesso tipo. In VHDL esiste la possibilità di utilizzare il record per raggruppare diversi elementi di tipi differenti in una sola struttura. Negli array per accedere ad un singolo componente basta specificare il suo numero tra parentesi dopo il nome della variabile; nei record le componenti, dette anche campi, non sono numerate ma hanno un nome esplicito. Per accedere ad una particolare componente di un record si usa la selezione: si scrive il nome della variabile, seguito da un punto e dal nome della componente a cui si vuole accedere. Il **listato 1** mostra una porzione di codice VHDL per un possibile esempio.

TIPI INTERI

Una variabile definita con l'uso di un tipo intero assume unicamente valori numerici. Con il tipo integer si possono rappresentare valori di 32 bit e può essere utilizzato per la sintesi: attenzione però nell'uso delle risorse disponibili.
Il valore che può assumere una variabile intera è comprensivo tra i due valori 2,147,483,647 e +2,147,483,647. In ogni caso, è necessario controllare, comunque, l'ambiente di sviluppo perché il valore dipende fortemente dall' implementazione. Esistono due sottotipi interi predefiniti:

- Natural, contiene valori numeri tra 0 e il massimo intero rappresentabile.
- Positive, contiene valori numerici tra 1 e il massimo intero rappresentabile.

FILE

Il linguaggio VHDL permette di interagire sui file di testo, in questo modo è possibile interagire con la console di simulazione. In realtà, è possibile anche utilizzare la parola chiave assert, anche se questo uso risulta meno flessibile del precedente.
Con assert è possibile segnalare la presenza di risultati discordi tra quelli attesi all'uscita di una entity con quelle prodotte, ma non il preciso vettore di test che ha prodotto l'errore. Il VHDL fornisce una serie di procedure standard per interagire con i file di testo ed è necessario specificarne l'uso mediante la dichiarazione del package text_io, come

```
use STD.textio.all; use ieee.std_log-
ic_textio.all;
```

IEEE

Può accadere, per permettere la sintesi, di specificare come valore di un segnale un valore diverso da 0 e da 1. La **tabella 3** mostra le funzioni di conversioni disponibili nella libreria IEEE. I tipi in esame utilizzano un sistema logico a 9 valori, descritti nella **tabella 4**. La libreria IEEE è divisa in una serie di package dove ognuno di questi definisce alcuni oggetti che possono essere utilizzati nella progettazione. Possiamo senz'altro citare il package std_logic_1164 che definisce i due tipi risolti std_logic e std_ulogic ed i corrispondenti tipi vettoriali std_logic_vector ed std_ulogic_vector. Per usare i tipi std_logic e std_logic_vector si deve prima rendere visibile il package ieee.std_logic_1164. Nel package std_logic_1164 oltre ad introdurre i tipi std_ulogic ed std_logic, si definiscono una serie di funzioni e procedure standard.

SOTTOTIPO

Un tipo definito come subtype (sottotipo) è un sottoinsieme di un tipo definito. Grazie a questa possibilità è possibile definire un oggetto con particolari restrizioni di un tipo base garantendo controllo più accurato dei vincoli imposti sui segnali e variabili. Il sottotipo eredita tutte le operazioni previste per il tipo base, oltre alla possibilità di definire ulteriori proprietà: il sottoinsieme che se ne ricava si applica solo ai valori. Il sottotipo si definisce mediante delle restrizioni:

```
subtype my_type  is  integer range -
1024 to +1024;
```

Un sottotipo può restringere I valori rappresentabili di un tipo scalare mediante l'uso di un intervallo dei valori specificabili. È possibile rappresentare anche un unconstrained array attraverso la specifica dei limiti per gli indici. Esistono anche due sottotipi numerici predefiniti:

```
subtype natural is integer range 0 to
highest_integer
subtype positive is integer range 1
to highest_integer
```

PHYSICAL TYPE

Il tipo physical type è utilizzato per rappresentare quantità fisiche del mondo reale, come lunghezza, massa, tempo, resistenza. La dichiarazione di una variabile di questo tipo include la definizione dell'unità di misura primaria che può essere eventualmente seguita da definizioni secondarie che sono multipli della fondamentale. In VHDL questo tipo predefinito è importante perchè è utilizzato per specificare I tempi in simulazione. E' necessario verificare che questa definizione è prevista nel proprio ambiente di sviluppo.

FLOATING-POINT TYPES (REAL)

Un tipo a virgola mobile (floating-point) è una approssimazione discreta dell'insieme di numeri reali in uno specifico intervallo di valori. Attraverso un tipo floating-point è possibile definire numeri reali con una mantissa e l'esponente. La precisione dell'approssimazione non è definita nello standard VHDL, ma l'unica cosa che viene definita è che deve essere almeno sei cifre decimali. L'intervallo dei valori deve essere almeno tra –1E38 e +1E38. E' opportuno consultare la documentazione di riferimento del sistema di sviluppo adottato.

TIPO ENUMERATIVO

Un tipo enumerativo è un insieme ordinato di caratteri o identificatori, come

```
type alu_function is (disable, pass,
add, subtract, multiply, divide);
```

TABELLA 4 – DEFINIZIONE DEL TIPO STD_ULOGIC DELLA LIBRERIA STANDARD IEEE 1164

SEGNALE	DESCRIZIONE
'0'	Valore logico 0
'1'	Valore logico 1.
'Z'	Consente di specificare una condizione di alta impedenza, in modo da poter sintetizzare circuiti tristate.
'-'	Consente di specificare condizioni dont'care, permettendo al sintetizzatore di ottimizzare il circuito, questa particolare condizione può essere solo assegnato e non generato in simulazione.
'H'	Segnale debole. Interpretabile come 1.
'L'	Segnale debole. Interpretabile come 0.
'W'	Segnale debole. Non è possibile interpretarlo come 0 o 1.
'U'	Indefinito. Il valore non è mai stato assegnato.
'X'	Valore indeterminato. Può essere generato da qualsiasi simulatore ogni volta che non è possibile determinare il valore di un segnale. Può essere 0 o 1.

ESERCIZI

ESERCIZIO 1

Rappresentate le seguenti stringhe:

a) Una stringa vuota

b) La frase 'ciao mondo'

ESERCIZIO 2

Bit string

a) rappresentare in ottale
la bit string seguente:
B"001_110_010"

b) rappresentare in binario
la bit string seguente:
x"0D"

c) rappresentare in valore binario
il seguente valore:4B

ESERCIZIO 3

Indicare se le seguenti istruzioni sono corrette:

a) x <= a + b

b) y <= '1'

c) y <= 1

d) c: bit_vector (15 downto 1)

e) b : bit_vector (3 to 0)

f) d : bit_vector (7 downto '0')

ESERCIZIO 4

Tipo definito dall'utente

rappresentare un valore intero in codifica binaria naturale su 5 bit si può definire un subtype del tipo integer con estremi 0 e 25-1=31

TABELLA 5 – ATTRIBUTI APPLICABILI ALL'OGGETTO SIGNAL

ATTRIBUTO	SIGNIFICATO
stable	Restituisce true se non c'è stato nessun evento su S nell'intervallo specificato.
event	Restituisce true se c'è un evento sul signal nel corrente ciclo di simulazione
last_event	Restituisce l'intervallo di tempo intercorso dall'ultimo evento
last_value	Restituisce il valore di S prima dell'ultimo evento

TABELLA 6 – ATTRIBUTI APPLICABILI AD UN ARRAY, ESEMPIO TYPE PROVA IS INTEGER RANGE 0 TO 255

ATTRIBUTO	SIGNIFICATO	VALORE
left	Restituisce l'indice sinistro della dimensione	0
right	Restituisce l'indice destro	255
High	Restituisce l'indice più grande	255
low	Restituisce l'indice più piccolo	0
range	Restituisce l'ampiezza	0 to 127
Reverse	Restituisce il range inverso	
Length	Restituisce la lunghezza del range	256
Ascending	Restituisce true se il range è ordinato in ordine crescente	

Gli elementi possono essere degli identificatori definiti dall'utente o singoli caratteri racchiusi tra apici. Il primo elemento è il più piccolo, mentre l'ultimo è il più grande. Nell'esempio precedente abbiamo: disable < pass < add < subtract.

TIPO DEFINITO DALL'UTENTE

Il programmatore VHDL può definire tipi propri, cioè tipi che ereditano le proprietà del tipo padre, magari dotate anche di proprie operazioni che devono essere ridefinite dal programmatore stesso. Per definire un tipo del genere è necessario seguire la seguente regola:

subtype new_type_name is type_name range val1 to val2;

cioè, con un esempio:

```
subtype nuovo_tipo is integer range 0
to 7;
```

In questo caso, il nuovo tipo è un sottoinsieme degli interi ed eredita le proprietà e le operazioni associate. I più comuni tipi definiti dall'utente sono i tipi enumerativi. a volte può risultare vantaggioso, se i tipi definiti dall'utente sono utilizzati in più progetti, raggruppare, oltre alle definizioni dei tipi, anche quelle delle definizioni di funzioni e procedure comuni in una libreria.

ATTRIBUTI

Un attributo serve ad estrarre da un oggetto informazioni che non sono disponibili attraverso i normali operatori. Possiamo senza dubbio mettere in evidenza due attributi associati ad un segnale, 'event e 'stable, che sono utilizzati per individuare i fronti del segnale di clock. Gli attributi iniziano tutti con il carattere apostrofo e si applicano alla destra del nome dell'oggetto (il prefisso). In questo modo ai segnali sono associati una serie di attributi che possono essere utilizzati all'interno del codice, le **tabella 5** e **6** mostrano il significato di alcuni attributi. E' opportuno consultare la documentazione tecnica per verificare gli attributi applicabili per ciascun tipo. ❏

CODICE MIP 2756767

Un generatore RF da 30MHz

Se vi serve un segnale radio per tarare il vostro ultimo ricevitore o volete capire se la vecchia radio a valvole del nonno ancora funziona, allora questo è il progettino che fa per voi; un piccolo ma efficiente generatore di radio frequenza realizzato con pochi componenti che non richiede l'avvolgimento di apposite bobine, capace di irradiare una portante RF modulabile in ampiezza per tutto lo spettro delle HF

Chi per hobby o per lavoro si occupa di radiotecnica, sa che uno degli strumenti indispensabili è proprio il generatore di radio frequenza, più comunemente conosciuto come oscillatore modulato.
Questo strumento oltre a permettere di testare e ad allineare il front-end dei ricevitori, consente di analizzare la risposta in frequenza dei filtri e degli amplificatori RF, di verificare la frequenza di risonanza dei circuiti accordati LC è insomma un valido aiuto ogni qualvolta è necessario avere a disposizione un segnale radio.

LA BANDA DELLE HF

Per semplicità il nostro generatore è stato concepito per la sola banda delle HF (High Frequency) (3 – 30 Mhz) più una porzione della banda delle MF (Medium Frequency) (0,3-3 Mhz).
La scelta è stata dettata da due semplici motivi:
1- La banda di frequenze è di valore relativamente basso e ciò permette la realizzazione di un circuito poco critico, assemblabile con comuni componenti attivi di facile reperibilità come transistor e FET.

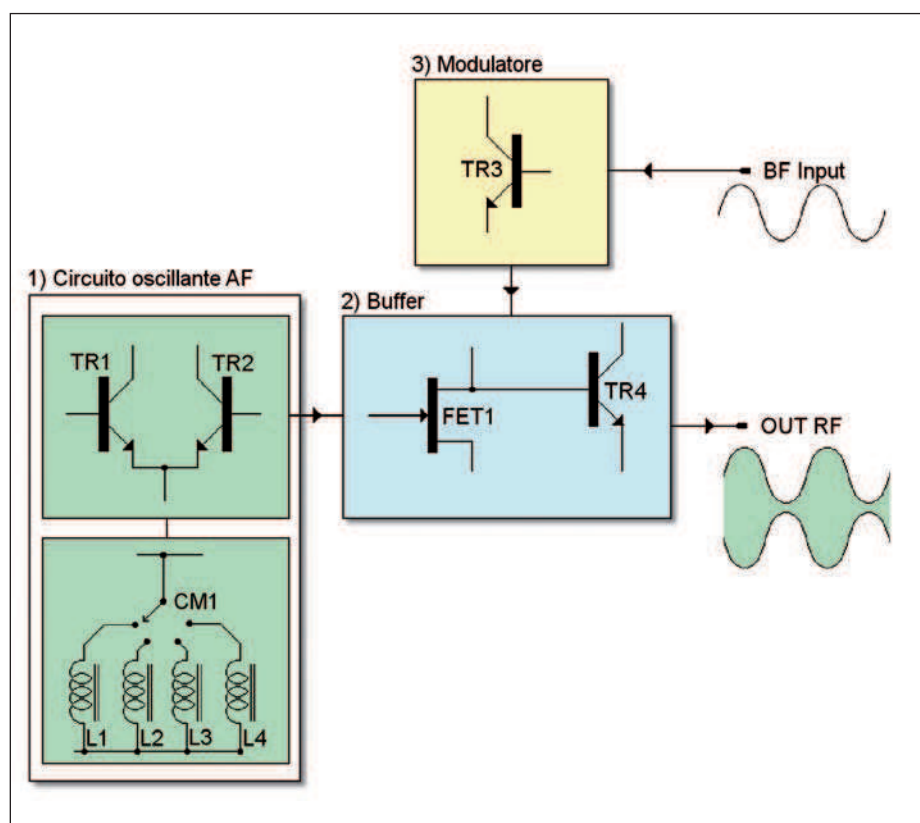



Figura 1: schema a blocchi del generatore RF.

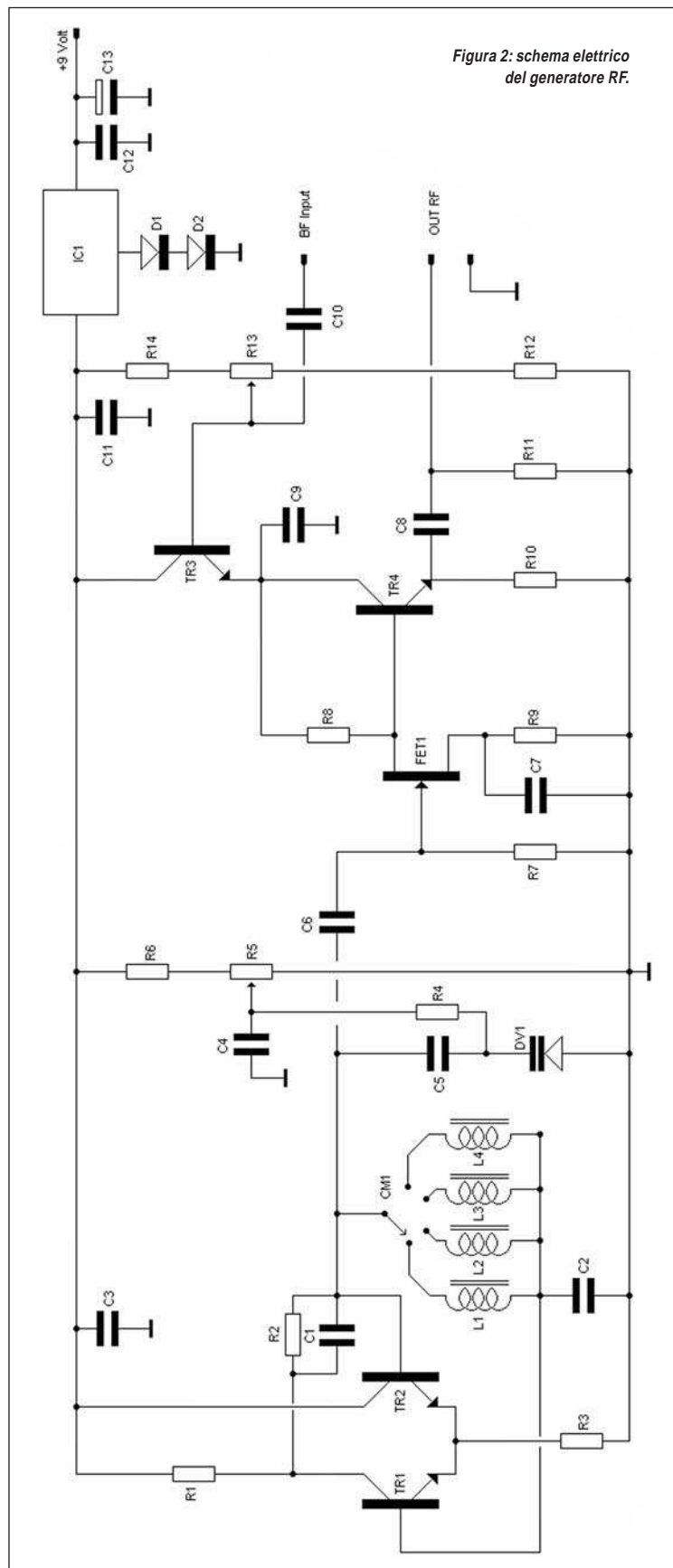



Figura 2: schema elettrico del generatore RF.

LISTA COMPONENTI

R1	1,2 kohm ¼ w	C1	12 pf ceramico, 63 V.	TR1,TR2	2N2222A
R2	120 kohm ¼ w	C2	100 nf ceramico, 63 V.	FET1	BF244A
R3	1,2 kohm ¼ w	C3	100 nf ceramico, 63 V.	TR3	2N706
R4	56 kohm ¼ w	C4	100 nf ceramico, 63 V.	TR4	BC550
R5	100 kohm pot. Lin.	C5	1,5 nf ceramico, 63 V.	IC1	78L05
R6	100 ohm ¼ w	C6	47 pf ceramico, 63 V.	D1,D2	1N4148
R7	100 kohm ¼ w	C7	10 nf ceramico, 63 V.	DV1	SVC 321 (vedi testo)
R8	1,2 kohm ¼ w	C8	560 pf ceramico, 63 V.	L1	47 uH (vedi testo)
R9	100 ohm ¼ w	C9	6,8 nf ceramico, 63 V.	L2	10 uH (vedi testo)
R10	120 ohm ¼ w	C10	100 nf ceramico, 63 V.	L3	3,3 uH (vedi testo)
R11	1,2 kohm ¼ w	C11	100 nf ceramico, 63 V.	L4	0,47 uH (vedi testo)
R12	1 kohm ¼ w	C12	100 nf ceramico, 63 V.		
R13	47 kohm trimmer vert.	C13	22 uf 25 volt elettrolitico		
R14	1 kohm ¼ w				

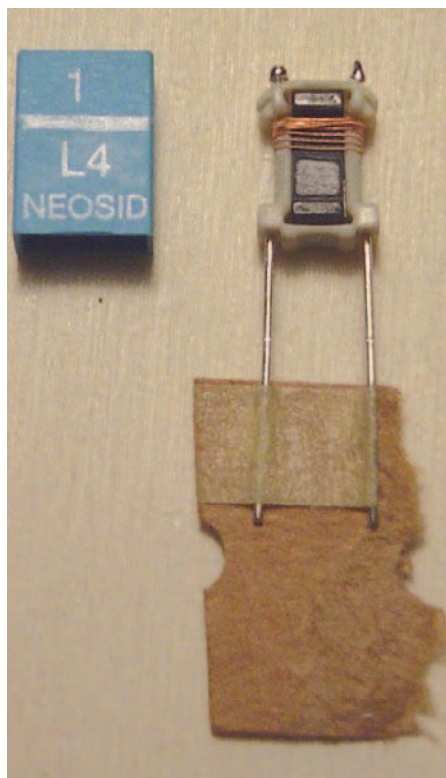



Figura 3: foto induttanza NEOSID.

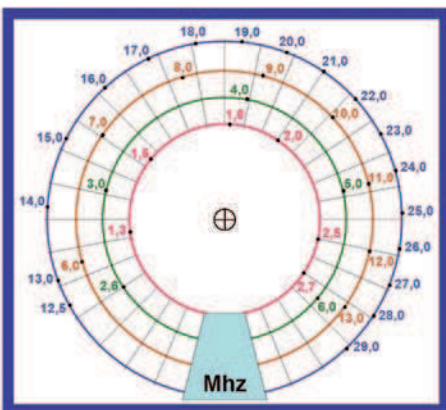
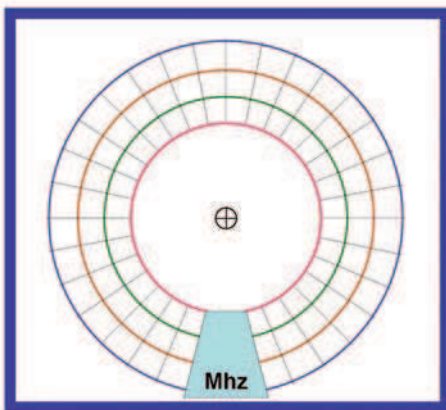



Figura 6: scala graduata vuota.

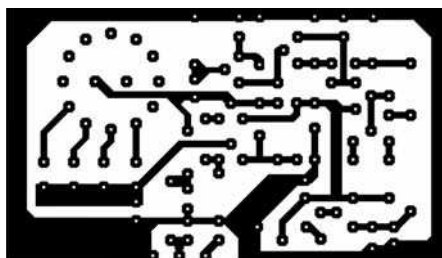



Figura 4: circuito stampato del generatore RF.

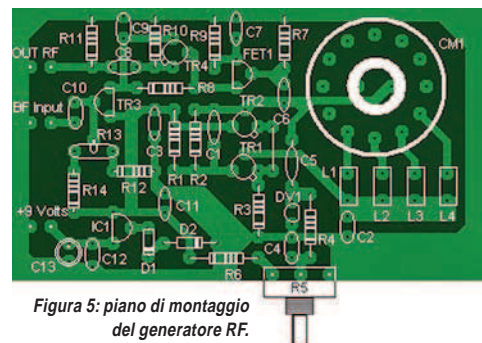



Figura 5: piano di montaggio del generatore RF.

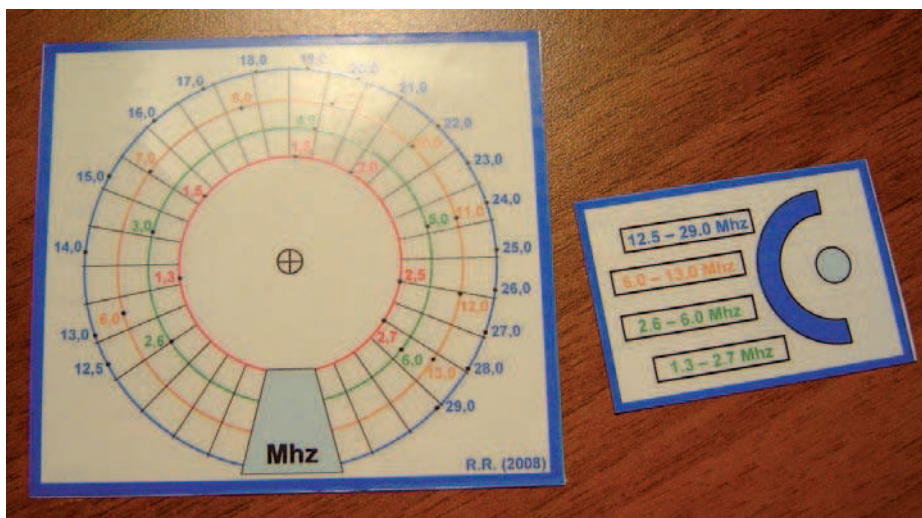



Figura 7: scala graduata finale.

2- La presenza di numerose frequenze assegnate ai radioamatori, permette di effettuare iteressanti esperimenti di ascolto con ricevitori auto costruiti e in tal caso il nostro generatore può essere d'ausilio nella messa a punto di tali apparecchiature.

In **tabella 1** si riportano le frequenze su cui operano i radioamatori nella banda delle MF/HF. Naturalmente oltre alle gamme radio amatoriali, si ha anche la possibilità di ascoltare un gran numero di stazioni commerciali sia Italiane che estere.

SCHEMA ELETTRICO

Cominciamo con l'analizzare lo schema a blocchi di **figura 1**, come si può vedere il nostro generatore è funzionalmente costituito da tre blocchi, che sono rispettivamente:

1- Un circuito oscillante in alta frequenza
2- Uno stadio buffer.
3- Uno stadio modulatore.

L'oscillatore AF è costituito dai transistor TR1 e TR2 in configurazione E.C.O (Emitter Couplet Oscillator). In pratica si tratta di un oscillatore in cui TR2 è configurato come amplificatore con uscita di emettitore e TR1 come amplificatore con base a massa. I due emettitori, come si può vedere dallo schema elettrico **(figura 2)**, sono collegati tra loro, mentre la piccola capacità C1 innesca l'oscillazione riportando il segnale in uscita dal collettore di TR1 sulla base di TR2. Le bande di frequenza su cui è possibile far funzionare l'oscillatore sono selezionabili tramite il commutatore CM1 e nell'ambito di ogni banda è possibile variare la frequenza tramite R5, il quale governa la polarizzazione del diodo varicap DV1. Il segnale RF generato da TR1/TR2 è applicato tramite il condensatore C6 al gate di FT1 che assieme a TR4 forma lo stadio buffer ovvero lo stadio che ha il compito di separare il circuito oscillante da quello che sarà poi il carico posto in uscita al circuito. In pratica, l'alta impedenza in ingresso del FET e la bassa impedenza in uscita sull'emettitore di TR4, configurato come "emitter-follower", fanno in modo che gli eventuali dispositivi applicati al generatore

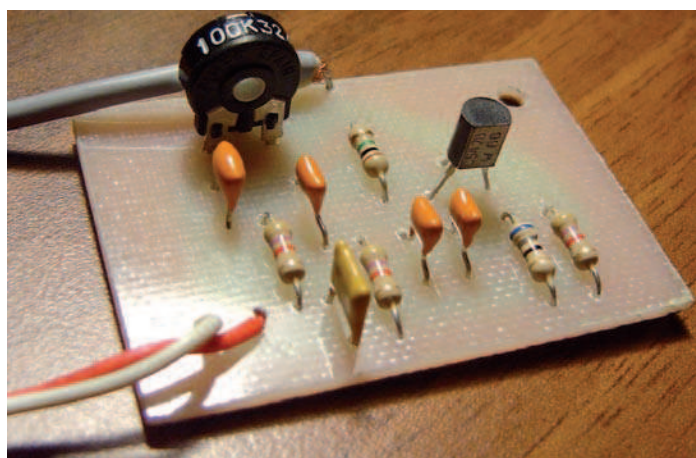



Figura 8: oscillatore BF sinusoidale.

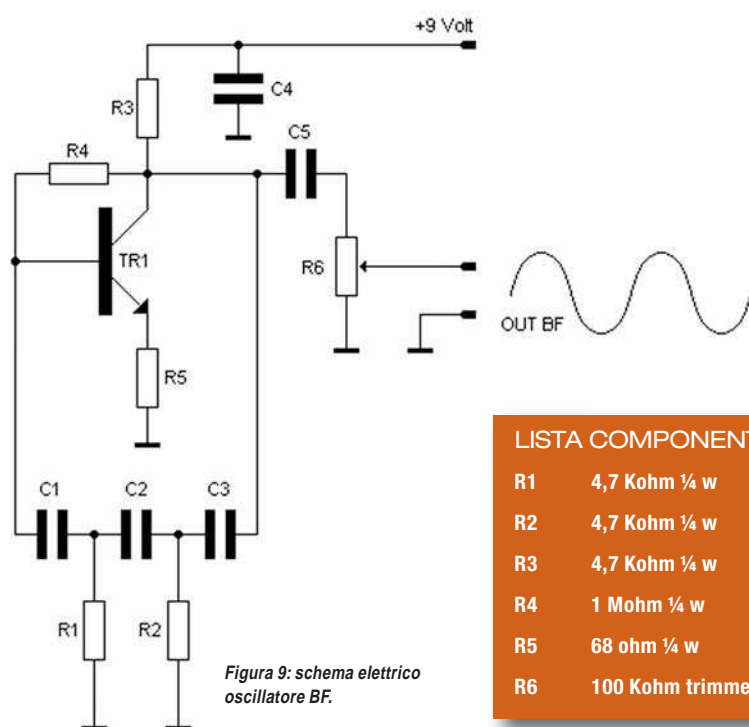



Figura 9: schema elettrico oscillatore BF.

RF non influenzino la frequenza di oscillazione di TR1/TR2. Sempre dallo schema elettrico è possibile vedere che il "tandem" FT1/TR4 è alimentato tramite un ulteriore transistor TR3 la cui polarizzazione è governa dal trimmer R13. TR3 in pratica è il nostro stadio modulatore; regolando opportunamente R13 e applicando un segnale audio al condensatore C10, faremo variare il punto di lavoro di TR3 e contemporaneamente varierà la tensione di alimentazione del gruppo FT1/TR4, in questo modo ai capi di R11 avremo disponibile una portante RF modulata in ampiezza. Il circuito è completato dallo stabilizzatore di tensione IC1 che con l'ausilio dei due diodi al silicio DS1 e DS2 alimenta il nostro generatore ad una tensione complessiva di circa 6,2 volt.

MONTAGGIO E TARATURA

Il montaggio del circuito non dovrebbe comportare particolari difficoltà, la maggior parte dei componenti è facilmente reperibile o sostituibile con equivalenti. Per

LISTA COMPONENTI

R1	4,7 Kohm ¼ w	C1	10 nf ceramico, 63 V.
R2	4,7 Kohm ¼ w	C2	10 nf ceramico, 63 V.
R3	4,7 Kohm ¼ w	C3	10 nf ceramico, 63 V.
R4	1 Mohm ¼ w	C4	100 nf ceramico, 63 V.
R5	68 ohm ¼ w	C5	100 nf ceramico, 63 V.
R6	100 Kohm trimmer vert.	TR1	BC547B

il diodo varicap, che nel nostro circuito, sostituisce il vecchio condensatore variabile, ho utilizzato un SVC 321 della SANYO equivalente al MVAM 115 della MOTOROLA, il quale garantisce un ampia variazione di capacità anche a basse tensioni di polarizzazione , eventuali possibili sostituzioni sono con il BB112, il BB509 o comunque con qualsiasi altro varicap di capacità minima di 30-40 PF e massima di 400-500 PF. Un discorso a parte va fatto per le induttanze che permettono il sezionamento dello spettro MF/HF in quattro sotto bande. Per semplicità ho usato delle comuni induttanze già avvolte della NEOSID **(figura 3)** ritoccando in base alla necessità il numero delle spire. I valori di partenza da utilizzare sono:

TABELLA 1: LE FREQUENZE SU CUI OPERANO I RADIOAMATORI NELLA BANDA DELLE MF/HF.

FREQUENZA	LUNGHEZZA D'ONDA		BANDA
1830-1850kHz	164-162,16 m	160m	Medium Frequency
3500-3800kHz	86- 78,95 m	80m	
7000-7100kHz	43- 42,25 m	40m	
10,1-10,2MHz	30- 29,55 m	30m	
14-14,3MHz	21- 20,91 m	20m	
14,3-14,4MHz	21- 20,91 m	20m	High Frequency
18,1-18,2MHz	17- 16,51 m	17m	
21-21,5MHz	14- 13,99 m	15m	
24,9-25MHz	12- 12,00 m	12m	
28-29,7MHz	11- 10,10 m	10m	

Figura 10: circuito stampato oscillatore BF.

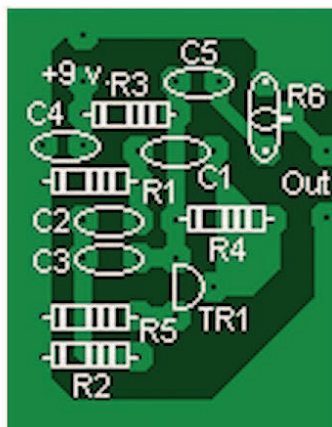



Figura 11: piano di montaggio oscillatore BF.

- 47 uH per la banda che va da 1 Mhz a 3 Mhz
- 10 uH per la banda che va da 3 Mhz a 6 Mhz
- 3,3 uH per la banda che va da 6 Mhz a 13 Mhz
- 0,47 uH per la banda che va da 13 Mhz a 30 Mhz

Per la realizzazione e la taratura si può procedere nel seguente modo:

1- Vanno Inserti tutti i componenti sul circuito stampato prendendo a riferimento il piano di montaggio di **figura 5**, senza inserire le quattro induttanze.

2- Date tensione al circuito e regolate il trimmer R13 fino a leggere, tra l'emettitore di TR3 e la massa circa 4-4,2 volt.

3- Partendo dalla banda di frequenze più basse (1-3 Mhz), inserite l'induttanza L1 sul circuito e collegate un frequenzimetro in uscita al generatore, agendo sul potenziometro R5 verificate la copertura di banda, qualora il limite inferiore sia molto al disotto di 1 Mhz, sfilate il contenitore plastico del componente e agendo sull'avvolgimento togliete 1 o due spire; ripetete questa operazione fino ad ottenere una copertura di banda prossima a quella stabilita.

4- Procedete in maniera analoga sulle altre tre induttanze.

Figura 12: traccia all'oscilloscopio del segnale RF modulato.

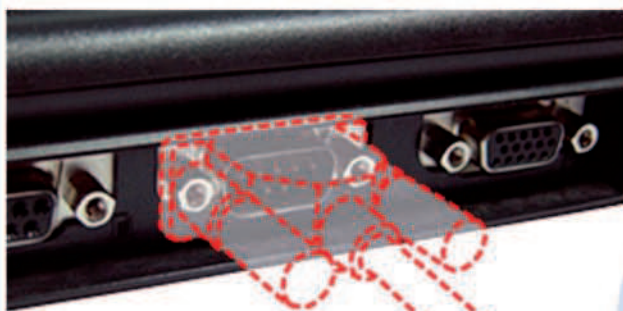
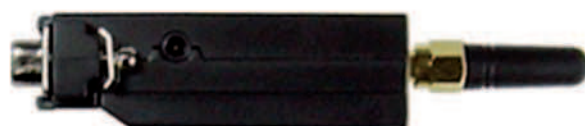
L'ampiezza del segnale prodotto dal generatore, non essendo previsti sistemi equalizzazione, varia in funzione della frequenza e va da circa 2 volt per le frequenze più basse a circa 0,5 -1,0 volt per quelle più alte. Per facilitare l'utilizzo dello strumento, senza dover ricorrere di volta in volta ad un frequenzimetro, consiglio la realizzazione di un'apposita scala graduata, del tipo visibile in **figura 6**. La realizzazione è abbastanza semplice; va stampata una scala vuota su carta semplice e va posizionata sul potenziometro R5, con l'ausilio di frequenzimetro vanno marcate con una matita le frequenze principali, dopo di che si riportato i valori rilevati sul computer e si stampa la scala graduata su carta adesiva trasparente **(figura 7)**.

Figura 13: prototipo.

MODULIAMO IN AMPIEZZA

Come possiamo modulare in ampiezza il nostro generatore RF? E' semplice, basta prendere un qualsiasi segnale audio (lettore MP3, lettore CD/DVD ecc.) di ampiezza massima pari a 1-1,5 Volt pp e applicarlo, tramite C10, alla base di TR3. Chi invece desidera modulare la portante radio con una segnale audio di frequenza fissa, può far riferimento allo schema di **figura 9** e al relativo piano di montaggio di figura 10/11, dove è riportato un oscillatore sinusoidale a sfasamento. In questo caso, l'oscillazione è ottenuta applicando ad un amplificatore invertente una serie di celle RC che sfasano il segnale di ulteriori 180° determinando quindi una reazione positiva tra l'ingresso e l'uscita. La frequenza generata è stabilita dalle resistenze R1, R2 e R3 e dai condensatori C1, C2 e C3, con i valori proposti si otterrà una sinusoide di frequenza compresa tra i 1000 e 1400 hertz. Il trimmer R1 permette di regolare l'ampiezza del segnale in uscita in maniera da non saturare lo stadio modulatore del generatore RF. La **figura 12** riporta quello che è possibile vedere all'oscilloscopio, modulando la portante RF con il segnale sinusoidale prodotto dal circuito sopra descritto. In **figura 13** si può vedere un prototipo in cui è stato messo assieme sia il generatore RF che l'oscillatore BF.

IN CONCLUSIONE

Vedrete che questo strumento non mancherà di essere utile in più occasioni, specialmente se avete una particolare passione per la radio. ❑

CODICE MIP 2756781

SATELLITI per *telecomunicazioni*

Il nostro pianeta è circondato da una vera e propria costellazione di satelliti che permettono di trasmettere e ricevere informazioni da ogni parte del globo, di seguire lo spostamento di mezzi mobili, di assistere la navigazione. Una rete globale che sta cambiando la nostra vita quotidiana e lo farà sempre di più nei prossimi anni

L'idea di utilizzare dei satelliti in orbita geostazionaria per le radiocomunicazioni, apparve per la prima volta in un articolo scritto da Arthur C. Clarke e pubblicato dalla rivista inglese "Wireless Word" nel 1945. In questo articolo, l'autore ipotizzò la messa in orbita sulla verticale dell'Equatore, di tre satelliti posti a 120° tra di loro e ruotanti alla stessa velocità angolare della terra, in modo da apparire fissi rispetto a qualsiasi punto di osservazione terrestre, realizzando così un possibile ponte radio tra due punti qualsiasi dell'emisfero.

I SATELLITI GEOSTAZIONARI

La caratteristica principale di un satellite geostazionario è quella di viaggiare alla stessa velocità angolare della terra, ovvero di essere geosincrono; questo vuol dire che il periodo del satellite è esattamente uguale alla durata del giorno sidereo (23 ore, 56 minuti e 4 secondi), ossia al tempo impiegato dalla Terra ad effettuare una rotazione completa attorno al proprio asse. Più esattamente, un satellite geosincrono si definisce geostazionario quando la sua orbita è circolare, si trova nello stesso piano dell'equatore terrestre, ed è diretta, ossia è descritta nello stesso verso di rotazione della terra. In tal caso il moto relativo fra satellite e superficie terrestre è nullo e quindi, ad un osservatore situato entro l'area di copertura, il satellite appare stazionario, ovvero immobile. La suddetta proprietà rende i satelliti insostituibili nelle comunicazioni radiotelefoniche ed in quelle televisive, infatti ciascuno di essi riesce a coprire con i suoi segnali quasi un terzo della superficie terrestre; con tre satelliti equidistanti in longitudine è quindi possibile garantire la copertura dell'intero pianeta.

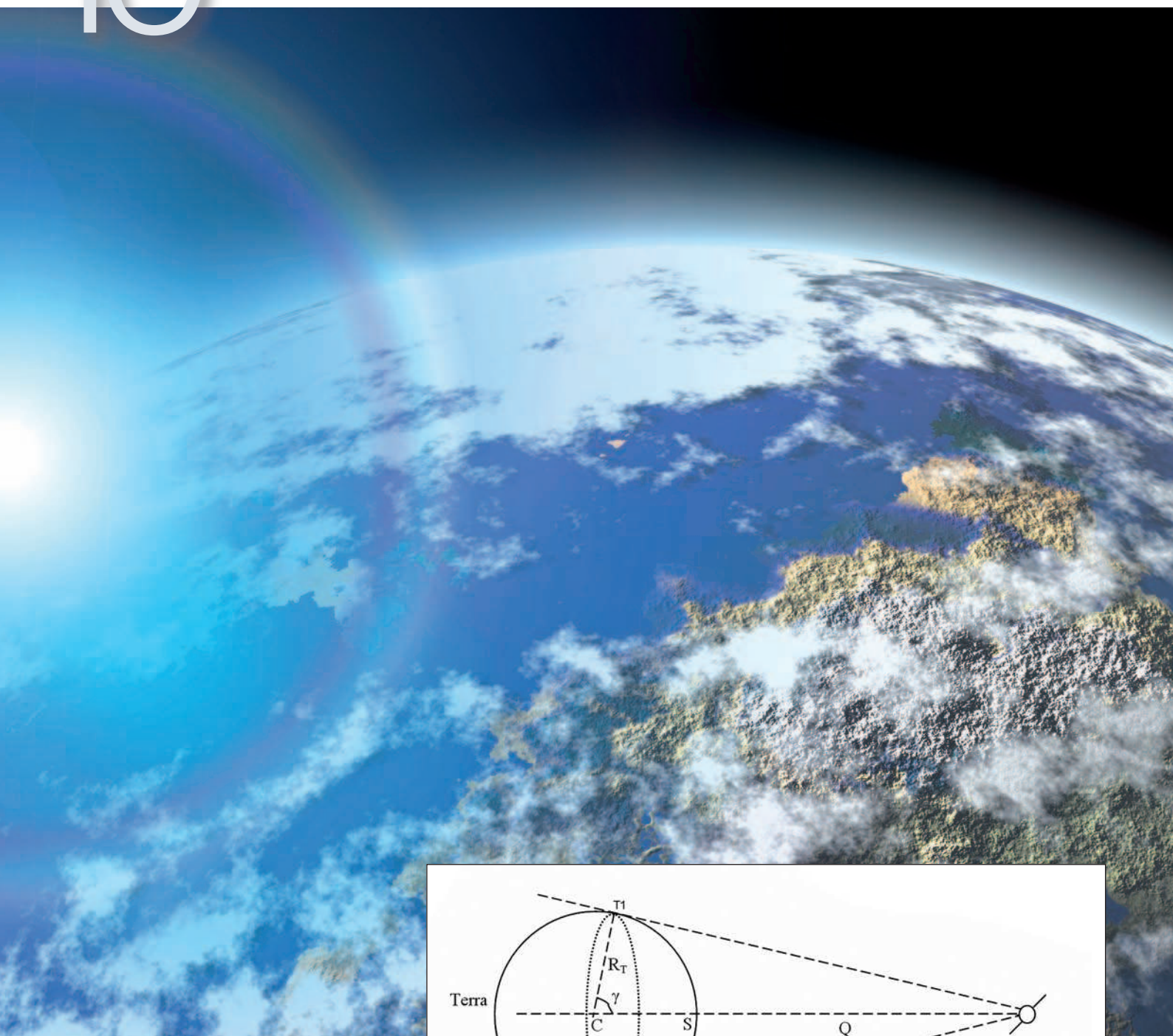



Figura 1: schematizzazione sul piano equatoriale del sistema terra-satellite.

Figura 2: schema di principio di un sottosistema di comunicazione.

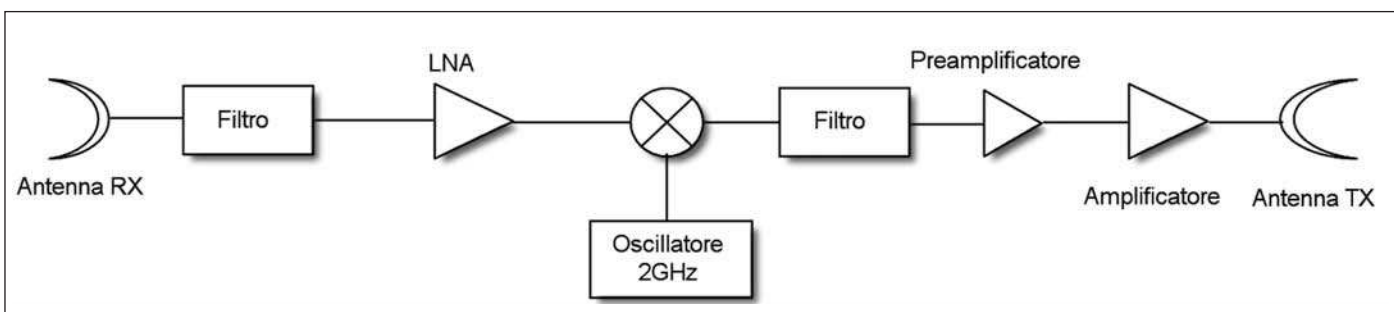
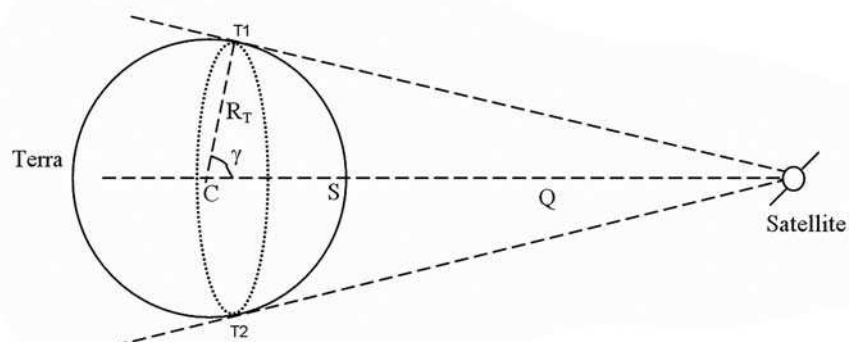
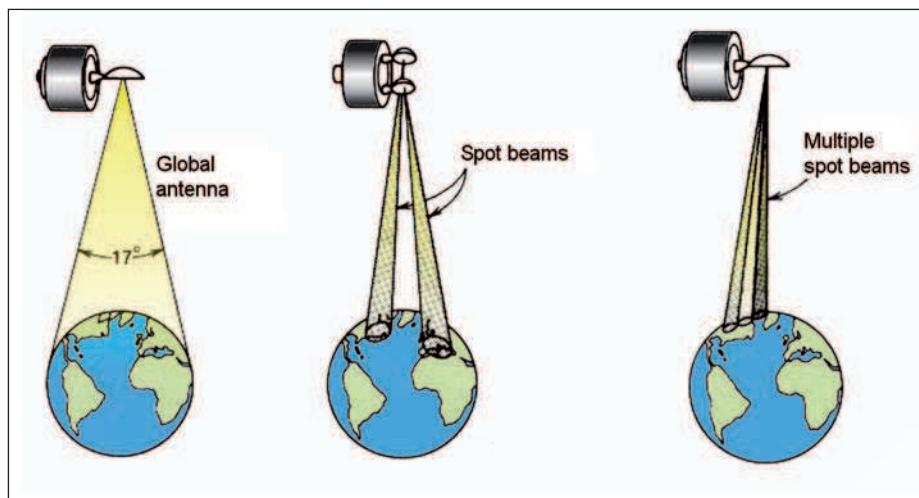



Figura 3: tipologie di "illuminazione" satellitare.

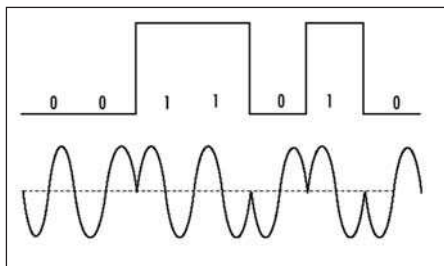



Figura 4: modulazione PSK.

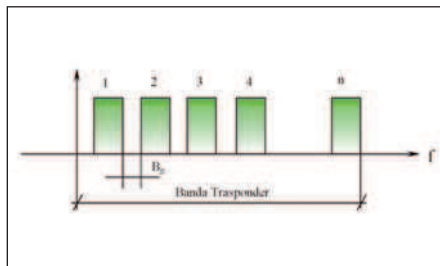



Figura 5: tecnica FDMA.

Per raggiungere e mantenere l'orbita geostazionaria, un satellite deve avere una quota di circa 42.000 Km dal centro della Terra (36.000 Km circa dall'Equatore) perché a questa distanza la forza centrifuga dovuta alla velocità del satellite e che permette un moto sincrono con la rotazione terrestre, eguaglia la forza di gravità della Terra. In quest'orbita il satellite ruota ad una velocità, pari a circa 6,6 volte la velocità di superficie della Terra all' Equatore, rimanendo così sincronizzato con la rotazione del globo.

L'AREA DI COPERTURA DI UN SATELLITE

Conducendo dal satellite le semirette tangenti alla sfera terrestre **(figura 1)**, su di essa s'individua un cerchio minore che rappresenta la linea dell'orizzonte geometrico del satellite. La porzione di superficie terrestre, racchiusa da tale linea, rappresenta il cosiddetto footprint, ossia l'area di copertura entro la quale si possono captare gli eventuali segnali trasmessi dal satellite. Nel caso dei satelliti geostazionari, il massimo angolo di visibilità tra la verticale del punto sub satellite S e uno dei punti di tangenza T_1 (o T_2) è pari a circa 81,3°. Con tale angolo è possibile definire geograficamente l'area di copertura. Le zone polari della Terra non sono pertanto osservabili dai satelliti geostazionari; analogamente, gli osservatori terrestri situati in tali aree non possono ricevere i segnali emessi dai satelliti. L'area utile, in realtà, è minore di quella teorica in quanto la ricezione dei segnali è molto più difficoltosa quando il satellite è in prossimità dell'orizzonte. In tal caso, infatti, le onde elettromagnetiche devono attraversare uno spessore di ionosfera e d'atmosfera maggiore rispetto a quello che si ha quando i segnali provengono da una direzione quasi perpendicolare alla superficie terrestre. Normalmente si considera un margine di copertura di 5°, per cui il raggio sferico dell'area di acquisizione si riduce dal valore teorico di 81,3° a circa 76°. In queste condizioni, teoricamente bastano solo 3 satelliti per coprire tutto il globo.
Inoltre, essendo immobile rispetto alla terra, questo tipo si satelliti non richiede nessun tipo di meccanismo di inseguimento delle antenne di terra.

Come fa un satellite a rimanere in orbita?
Anche se il moto dei satelliti geostazionari è governato da complessi sistemi di equazioni matematiche, in realtà, l'applicazione di alcuni semplici leggi fisiche, ci permette di ricavare i valori dei fondamentali dei parametri orbitali (distanza dalla terra e velocità di rotazione).
Indichiamo con:
m = massa del satellite
r = distanza del satellite dal centro della terra
ω = velocità angolare del satellite
M = massa della terra
G = costante di gravitazione universale
Supponendo l'orbita perfettamente circolare e uniforme, il satellite è soggetto alla forza centrifuga radiale (in modulo):

$$m \cdot \omega^2 \cdot r$$

Alla stessa distanza r la forza d'attrazione gravitazionale deve essere uguale in modulo alla forza centrifuga per mantenere il satellite in equilibrio:

$$m \cdot \omega^2 \cdot r = G \cdot \frac{M \cdot m}{r^2}$$

da cui si ricava :

$$r^3 = \frac{G \cdot M}{\omega^2}$$

La velocità angolare del satellite vale:

$$\omega = \frac{2 \cdot \pi}{T}$$

dove:
T = periodo di rotazione del satellite.
Imponendo che il periodo T del satellite sia uguale a quello di rivoluzione della terra:
T= giorno sidereo = 86164 sec (è diverso dal giorno solare medio =24h x 60m x 60s= 86400 sec)

$$\omega = \frac{2 \cdot \pi}{T} = \frac{2 \cdot 3.14}{86164} = 7.29 \cdot 10^{-5} \, rad/\sec$$

per cui:

$$r = \sqrt[3]{\frac{G \cdot M}{\omega^2}} = \sqrt[3]{\frac{6.67 \cdot 10^{-11} \cdot 5.98 \cdot 10^{24}}{\left(7.29 \cdot 10^{-5}\right)^2}} \cong 42.000$$

Ricordando che r è la distanza del satellite dal centro della terra (in km), dob-

biamo sottrarre il raggio della terra R ≈6000 Km;
h = distanza del satellite dalla superficie terrestre = r - R = 42000-6000≈ 36000 km
La velocità del satellite è quindi:

$$v = \omega \cdot r = 7.29 \cdot 10^{-5} \cdot 42000 \cong 3.1 Km/\sec$$

Come si vede la velocità orbitale non dipende dalla massa del satellite; il solo limite per cui non si mandano in orbita grandi satelliti è legato alla difficoltà di lancio!

ANATOMIA DI UN SATELLITE

Un satellite è composto di molte parti, che variano a seconda del tipo di missione a cui è destinato; due elementi sono tuttavia comuni a tutte le tipologie e sono denominati tecnicamente "payload" (carico utile) e "bus" (il cosiddetto velivolo). Il carico utile è l'insieme di tutti gli equipaggiamenti di cui il satellite necessita per svolgere le sue funzioni e varia a seconda della applicazione: per esempio, in un satellite per telecomunicazioni, il payload è rappresentato dal cosiddetto "transponder" o ripetitore. Il bus è invece quella parte del satellite che porta il carico utile e tutti i suoi apparati nello spazio. Ha il compito di tenere strutturalmente unite tutte le parti del satellite e di garantire energia elettrica, propulsione, elaborazione dati etc. La struttura meccanica del bus è realizzata in leghe metalliche leggere ad alta resistenza oppure con materiali speciali a base di fibra di carbonio. La struttura deve essere molto robusta per resistere alle fortissime accelerazioni subite dal satellite durante la fase di lancio. Sul bus sono montati vari sottosistemi: controllo dell'orbita e dell'assetto, telemetria e comando, alimentazione, comunicazione, controllo termico, elaborazione dati. Vediamo ora le loro caratteristiche.

Sottosistema di controllo dell'orbita e dell'assetto
Ha il compito di mantenere il satellite sulla giusta orbita, compensando eventuali perturbazioni e provvede a mantenere le antenne ed i pannelli solari orientati nella giusta direzione. Per quanto concerne il controllo orbitale, le azioni sono finalizzate a regolare la velocità lungo la direzione della traiettoria, la quota dalla superficie terrestre, l'inclinazione, la posizione reciproca fra satelliti geostazionari. Il sottosistema di Propulsione è costituito di un serbatoio di propellente liquido (normalmente Idrazina), da una rete idraulica per la distribuzione dello stesso e di vari motori, in grado di "bruciare" ed espellere piccole quantità, accuratamente controllate,

LE TAPPE FONDAMENTALI DELLE COMUNICAZIONI SATELLITARI.

ANNO	EVENTO	CARATTERISTICHE
04/10/1957	**Lancio dello Sputnik (URSS)**	**Trasmetteva un segnale di identificazione in codice morse sulle frequenze di 20 e 40 Mhz**
03/11/1957	**Lancio del satellite Grande Sputnik (URSS)**	**Aveva caratteristiche orbitali uguali al precedente ma un peso superiore di circa 6 volte (510Kg). Girò intorno alla Terra diverse migliaia di volte fino al 14/10/1958.**
1958	**Lancio del satellite EXPLORER 1 (USA)**	**Il sistema di ricezione a terra operava a 150 Mhz**
1958	**SCORE (USA)**	**Inizio dell'era della radiodiffusione da satellite**
1963	**Lancio del satellite SYNCOM 2**	**Primo satellite geostazionario. Disponeva di una capacità di 300 circuiti telefonici o un canale televisivo**
20/08/1964	**Nascita INTELSAT**	**Inizio dell'era dei satelliti commerciali: 11 paesi firmarono l'adesione all'INTELSAT (International Telecommunications Satellite Organization) il primo sistema di telecomunicazioni globale**
1965	**Lancio del satellite INTELSAT I**	**Primo satellite commerciale geostazionario. Operava su tratte transatlantiche**
1977	**Conferenza amministrativa mondiale WARC 77**	**A Ginevra sotto l'egida dell'UIT (Unione Internazionale delle Telecomunicazioni), venne pianificato il servizio di diffusione diretta da satellite dei segnali televisivi e radiofonici DBS (DIRECT BROADCASTING SATELLITE).**
1980	**Lancio del satellite INTELSAT V**	**Tali satelliti lavorano sulle bande "C" e "KU" (frequenza che oggi viene usata per le nostre trasmissioni Satellitari)**

Dal 1957 ad oggi sono stati messi in orbita migliaia di satelliti per comunicazioni, navigazioni, per ricognizione fotografica, di tipo scientifico e di tipo meteorologico.

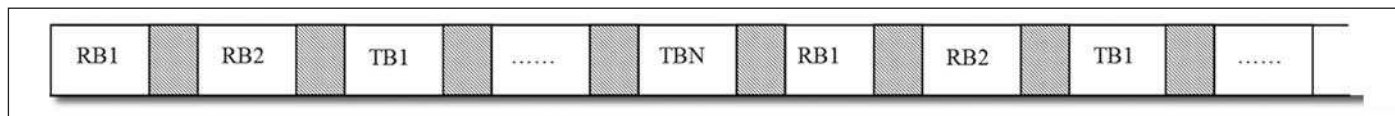



Figura 6: struttura di un Frame.

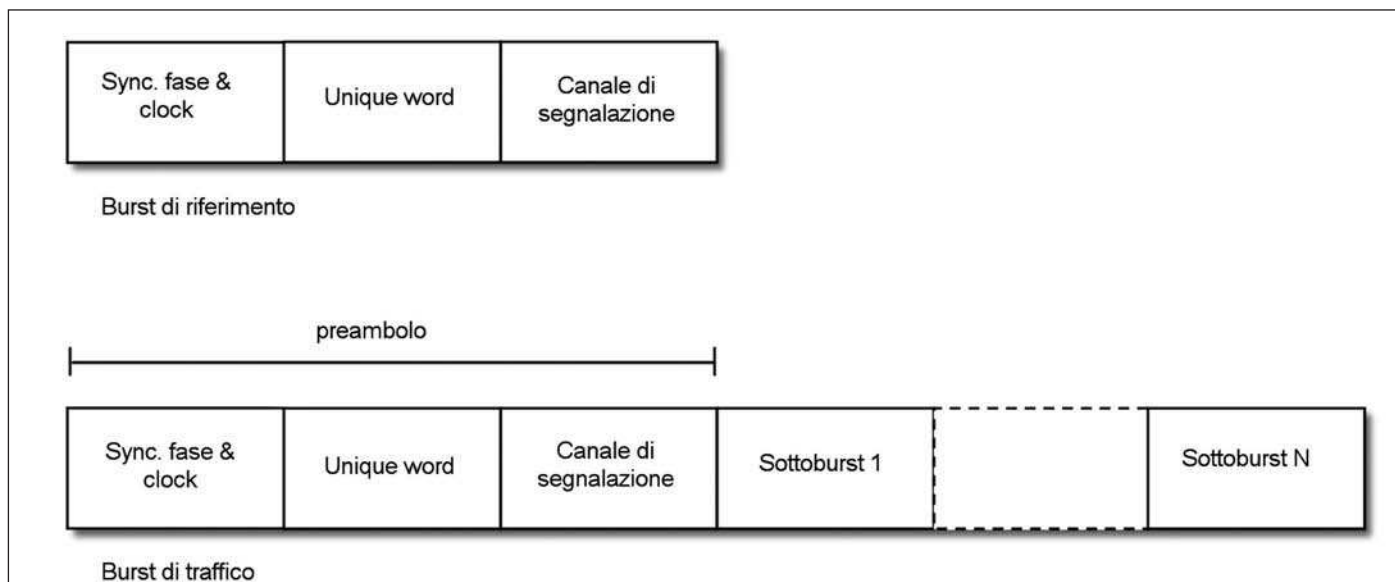



Figura 7: struttura dei Burst di Riferimento e di Traffico.

di propellente. Il controllo dell'assetto viene effettuato attraverso l'applicazione di momenti: cioè non si applicano accelerazioni lineari ma accelerazioni angolari. Ci sono diversi modi per applicare dei momenti ad un corpo satellitare. Il primo è quello in cui al satellite, che possiamo immaginare come un cilindro, viene applicato un momento, in modo tale da ottenere il cosiddetto effetto giroscopico: in questo modo il satellite è forzato a mantenere un orientamento e ad essere insensibile alle variazioni del campo di forze gravitazionale. Con questo sistema, il satellite ruota costantemente ad una velocità angolare dai 30÷100 rad/min lungo una certa direzione. Un altro modo di applicare momenti è quello che prevede l'uso delle cosiddette ruote di momento, cioè di ruote applicate in delle cavità sigillate e sottovuoto: l'applicazione di una accelerazione ad una di queste 3 ruote (poste lungo i tre assi ortogonali) per il principio di conservazione dei momenti angolari, provoca la rotazione del corpo del satellite nella direzione voluta. Questa situazione prende il nome di controllo sui tre assi.

Sottosistema di telemetria, tracking e comando

Si occupa della trasmissione dei dati rilevati da una serie di sensori posti a bordo del satellite. Esegue inoltre il controllo della corretta orbita dal satellite tramite un "inseguimento" da terra. Sulla base dei dati ricevuti ed elaborati da un computer di terra, vengono emessi dei comandi, che vengono poi eseguiti a bordo del satellite.

Sottosistema di alimentazione

Per la quasi totalità dei sistemi satellitari, l'energia solare, convertita in elettricità dalle celle fotovoltaiche, è la sorgente primaria di potenza a bordo. Vi sono dei pannelli di celle solari posti sul corpo del satellite nel caso dei sistemi "spinner", mentre nel caso di sistemi stabilizzati sui tre assi, si fa ricorso a delle lastre che supportano celle solari e che vengono ruotate nell'arco delle 24 ore in modo che ci sia sempre incidenza normale della radiazione sulle lastre stesse. La radiazione solare all'altitudine geostazionaria è di circa 1,3÷1,4 Kw/m^2; l'efficienza di conversione che si realizza è del 10÷15%. Naturalmente sia nelle fasi di lancio che durante le eclissi subentrano in funzione dei set di batterie.

Sottosistema di comunicazione

La parte principale del sistema di comunicazione è il transponder **(figura 2)**. Nella sua versione più essenziale, cioè nei sistemi non rigenerativi, esso si compone di un'antenna ad alto guadagno, che riceve il segnale trasmesso da una stazione terrestre; di un filtro a radiofrequenza che, avendo preso come esempio un sistema a 6 / 4 Ghz è centrato sui 6 Ghz. Seguono di un amplificatore a basso livello di rumore ("Low Noise Amplifier", LNA) e un convertitore di frequenza (normalmente la frequenza di ricezione è maggiore di quella di trasmissione, quindi si parla di un "down-converter"). Avremo poi un filtro centrato sui 4 Ghz per eliminare le frequenze non volute che vengono generate dall'operazione di conversione nel mixer. Abbiamo infine un pre-amplificatore RF, un amplificatore di potenza e un'antenna ad alto guadagno in trasmissione. Il sottosistema d'antenna influenza di molto la capacità, sia in termini di copertura che di potenza e di gestione di un sistema satellitare. Ci possono essere diversi tipi di illuminazione **(figura 3)**: la cosiddetta global beam, che è quella a più larga copertura (con una larghezza di fascio di circa 17° e tale che dall'altitudine geostazionaria riesce a coprire tutta la superficie terrestre vista); poi c'è la spot beam (copertura di zona) in cui un fascio riesce a coprire più di una

stazione di terra; c'è infine il sistema "multiple spot beam" un tipo di illuminazione che interessa le ultime generazioni di ripetitori satellitari che sono i sistemi più efficienti da un punto di vista spettrale e in cui il singolo fascio è orientabile dall'antenna stessa. Le caratteristiche geometriche dell'antenna ad apertura e la frequenza di lavoro hanno una diretta incidenza sul guadagno e sugli angoli di fascio; ricordiamo che il guadagno è un parametro che misura la capacità dell'antenna di convogliare la potenza verso una particolare direzione. In particolare, all'aumentare della frequenza e delle dimensioni geometriche dell'antenna, i fasci di illuminazione diventano sempre più stretti e direttivi.

Sottosistema di Controllo Termico

Svolge l'importante funzione di mantenere la temperatura del satellite entro limiti accettabili. Un satellite in orbita è esposto a temperature estreme: dai -120 gradi della parte in ombra ai 180 gradi sopra lo zero della parte esposta al sole, un'escursione termica di 300 (o più) gradi centigradi! Questo sottosistema utilizza riscaldatori (controllo attivo) e vernici, superfici riflettenti, "coperte termiche" per mantenere la temperatura degli apparati elettronici entro un'escursione più contenuta (tipicamente da - 20 a +50 gradi centigradi).

Sottosistema di Elaborazione Dati

E' il "cervello" di tutto il satellite. E' composto di un computer centrale, di una serie di programmi software molto complessi e di varie unità d'interfaccia verso gli altri sottosistemi del satellite.

LE STAZIONI DI TERRA

I collegamenti tra il satellite e terra avvengono normalmente in bande di frequenza sull'ordine dei GHz, in modo da perforare la ionosfera ed evitare interferenze causate da fenomeni atmosferici. Nelle diverse direzioni di trasmissione vengono usate portanti di frequenza diversa per evitare interferenze; ad esempio si userà sulla tratta di up-link (terra-satellite) una portante a 6 GHz e su quella di down-link (satellite-terra) a 4 GHz. La stazione di terra ricevente e' costituita essenzialmente da 3 blocchi:

- preamplificatore LNA;
- convertitore a frequenza intermedia in modo da poter elaborare il segnale captato con apparecchiature meno complesse (che lavorino cioè a frequenze più basse, intermedie);
- demodulatore (numerico o analogico), che riporta il segnale captato in una forma utile per essere compreso dai computer e dagli apparecchi elettronici.

Grande risalto è assunto dal preamplificatore a basso rumore. Infatti, essendo il satellite molto lontano dalla stazione ricevente e considerata la bassa potenza disponibile (dell'ordine delle decine di watt, limitata dal peso del satellite e dalla scarsa energia disponibile a bordo), l'intensità del segnale ricevuto a terra è molto bassa. In queste condizioni è molto importante che l'antenna ricevente sia di maggiore diametro possibile e che il primo stadio della stazione ricevente, che ha lo scopo di amplificare questo segnale, non contribuisca all'aumento della temperatura di rumore; ciò infatti porterebbe il rapporto segnale/rumore (SNR) sotto i livelli accettabili. Per risolvere questo rilevante problema basta tenere in considerazione due fattori e cercare di controllarli il più possibile:

- il rumore elettronico generato dalle apparecchiature dello stadio d'ingresso dell'antenna di terra;
- il rumore captato dall'antenna.

Di solito il primo tipo di disturbo è limitato mantenendo gli apparati elettronici ad una temperatura molto prossima allo 0 assoluto (in gradi Kelvin!); mentre il secondo problema può essere minimizzato con un attento posizionamento dell'antenna ricevente;

COME VENGONO MESSI IN ORBITA I SATELLITI?

Per portare un satellite in orbita geostazionaria, il vettore deve essere accelerato ad una velocità di circa 3000 m/s, (in modo da poter sfuggire al campo gravitazionale), deve essere posto in orbita in condizioni di inclinazione zero (deve stare cioè sul piano equatoriale); deve raggiungere infine la quota di circa 42.000 Km dal centro della terra. Esistono due tecniche di lancio principali:

1) ELV- Expendable Launch Veichles-basata sull'uso di razzi vettori non riutilizzabili;
2) STS-Space Transpostation Sistem-basata su razzi vettori riutilizzabili.

Il sistema ELV (utilizzato per esempio dai vettori Delta e Ariane), opera la messa in orbita in due step successivi: si pone dapprima il satellite in un'orbita ellittica in un piano inclinato rispetto al piano equatoriale e successivamente viene accelerato, per porlo in orbita circolare ad inclinazione zero rispetto al piano equatoriale.

Arthur C. Clarke

Arthur C. Clarke, l'ideatore delle comunicazioni satellitari, è nato a Minehead, (Inghilterra) il 16 Dicembre 1917, ha al suo attivo una produzione scientifico letteraria assai estesa, ma è ai più noto per il suo romanzo "2001 Odissea nello Spazio" da cui è stato sceneggiato l'omonimo film del regista Stanley Kubrick. In onore dello scrittore, la distanza dell'orbita in cui stazionano i satelliti geostazionari è denominata oggi "Fascia di Clarke".

Nella tecnica STS invece lo Space Shuttle viaggia in un'orbita equatoriale ad una quota di circa 296 Km; da quest'orbita lancia il satellite nell'orbita di trasferimento, dopodiché un altro motore immetterà il satellite stesso nell'orbita finale.

LINK SATELLITARI E MODULAZIONI

Nelle trasmissioni satellitari, l'elevata distanza del collegamento (36000 Km!), comporta notevoli problemi che derivano dalla grande attenuazione dei segnali trasmessi (che si smorzano con il quadrato della distanza percorsa) e dal ritardo di propagazione. La necessità di elevate potenze di trasmissione porta inoltre ad utilizzare gli amplificatori di potenza al limite della zona di saturazione, con conseguente introduzione di fenomeni di non linearità e distorsioni sul segnale trasmesso.Questo in generale produce due effetti: allargamento dello spettro rispetto agli amplificatori di potenza "tradizionali" funzionanti in zona di linearità e introduzione di interferenze intersimboliche che possono pregiudicare la qualità del segnale. Mentre nei ponti radio terrestri si tende all'uso delle modulazioni numeriche di tipo QAM (Quadrature Amplitude Modulation), nei collegamenti satellitari per i motivi sopra indicati, si utilizzano prevalentemente modulazioni numeriche di fase di tipo PSK (Phase Shift Keyng) e MSK (Minimum Shift Key). Nella modulazione PSK l'informazione è associata ad una variazione di fase della portante. In **figura 4** è riportato un esempio. La gamma di frequenze satellitari si estende dalla banda P alla banda Ka e oltre:

- banda P (0.2-1 GHz)
- banda C (4-8 GHz)
- banda Ku (10-18 GHz)
- banda Ka (18-31 GHz)

Come regola generale più alta è la frequenza e più estesa è la banda e la capacità del collegamento; per contro, più la frequenza è alta e più alta sarà l'attenuazione indotta dall'atmosfera e il rumore che si sommerà al segnale.

TECNICHE SATELLITARI DI ACCESSO MULTIPLO

Quando si parla di tecniche di accesso multiplo, ci si riferisce sostanzialmente ai vari modi con cui le risorse che ha un transponder, ossia la banda di frequenza disponibile e la potenza a bordo, vengono suddivise tra più utenti. Ci sono vari modi per classificare i sistemi di accesso: il criterio più tipico è quello di contraddistinguerli in base al modo in cui vengono suddivise le risorse (frequenza, tempo, codice). In questa ottica esaminiamo di seguito i principali sistemi di accesso e le loro caratteristiche.

Tecnica FDMA

Consente di gestire trasmissioni di tipo analogico, in divisione di frequenza; la sigla FDMA sta per Frequency Division Multiple Access: ogni stazione di terra ha assegnata una determinata banda e quindi, l'intero spettro a disposizione del transponder viene suddiviso tra più utenti; con questa tecnica, ogni utente, trasmette per tutto il tempo a disposizione, cioè non si hanno conflitti di accesso alla stessa risorsa, utilizzando però solo una parte della banda complessiva a disposizione.

In **figura 5** è riportato un esempio di ripartizione della banda del trasponder, suddivisa in n sottobande, ciascuna delle quali è assegnata ad un determinato collegamento. Sono previste delle "bande di guardia" per la minimizzazione delle interferenze tra i segnali relativi a canali adiacenti, dovute alla non perfetta limitazione della banda dei segnali stessi e alla possibilità di derive in frequenza degli oscillatori impiegati nei relativi sistemi di comunicazione. La stazione ricevente a Terra seleziona una sottobanda o una sua porzione attraverso un filtraggio selettivo effettuato sul segnale a radiofrequenza o a frequenza intermedia. Prescindendo dal controllo sulle bande occupate dei segnali, in un sistema FDMA non è necessario un vero e proprio coordinamento delle stazioni di Terra che accedono al satellite, a differenza di quanto accade con le altre tecniche di accesso multiplo. Una tipica applicazione satellitare FDMA è la trasmissione di segnali telefonici e televisivi analogici, su portanti modulate in frequenza.

Tecnica TDMA

Nell'architettura TDMA (Time Division Multiple Access), ogni utente ha invece a disposizione tutta la banda, ma non per tutto il tempo, quindi la trasmissione si organizza in trame: ogni trama viene suddivisa in un certo numero di intervalli temporali, ed ognuno di questi è destinato un particolare utente. Nel sistema TDMA, l'intera larghezza di banda è messa a disposizione di ogni utente ma su una base time-sharing. In questo caso è ovviamente necessario prevedere un sistema di sincronizzazione delle stazioni che accedono al satellite in modo da evita-

PER *approfondire...*

http://it.kingofsat.net/satellites.php – calcolo delle orbite dei satelliti
www.esa.int – sito web dell'agenzia speziale europea
http://it.wikipedia.org/wiki/Arthur_C._Clarke - Breve biografia di A. Clarke
http://it.wikipedia.org/wiki/Rete_satellitare -

re che più di una di esse tenti di utilizzarlo simultaneamente, dando luogo al cosiddetto fenomeno della "collisione" dei segnali trasmessi. A ciascuna stazione viene infatti riservato periodicamente un intervallo di tempo durante il quale essa trasmette una sequenza opportuna di dati, detta "burst". Gli istanti di trasmissione dei burst sono controllati opportunamente, in modo che essi arrivino al satellite senza sovrapporsi. In fase di ricezione, ciascuna delle stazioni servite dal satellite può ricevere l'intera sequenza di burst, dalla quale essa estrae solamente quelli di sua competenza. La sequenza di burst viene ripetuta ciclicamente secondo una organizzazione temporale denominata "frame" (o trama), la struttura del quale è illustrata schematicamente in **figura 6**. Il frame è costituito generalmente da: due "burst di riferimento" RB1 e RB2, da alcuni "burst di traffico" e da intervalli di separazione dei burst, detti "intervalli di guardia". Gli intervalli di guardia sono necessari per evitare sovrapposizioni dei burst relativi alle diverse stazioni di Terra. RB1 viene detto "burst di riferimento primario" e viene trasmesso da una specifica stazione della rete, designata come la stazione di riferimento primario. RB2 è detto "burst di riferimento secondario" e viene trasmesso da un'altra stazione, detta di riferimento secondario, la quale si sostituisce automaticamente alla prima quando essa va fuori servizio, in modo da garantire la continuità del funzionamento per la rete TDMA. I burst di riferimento non trasportano traffico in quanto essi hanno funzioni di controllo, di servizio e di sincronizzazione; in particolare, essi vengono utilizzati per la corretta sequenzializzazione dei burst di traffico che accedono al trasponditore. I burst di traffico vengono trasmessi dalle stazioni di traffico e trasportano i dati. Ciascuna delle stazioni può inviare una o più burst di traffico per ciascun frame TDMA, posizionati entro il frame stesso in opportuna relazione temporale rispetto ai burst provenienti dalle altre stazioni. La lunghezza di un burst di traffico dipende dalla quantità di informazione che esso deve trasportare e può in certi casi essere modificata, quando necessario. Nel burst di traffico, i bit informativi (detti di traffico) sono normalmente preceduti da un gruppo di bit, detto "preambolo", che ha funzioni di sincronizzazione, di identificazione e di trasporto di informazioni utili alla gestione e al controllo del sistema. I burst di riferimento sono invece costituiti dal solo preambolo. Di solito il preambolo consta di tre parti distinte: la sequenza di sincronizzazione di portante e di clock (CCR, carrier and clock recovery), la "parola unica" (UW, unique word) ed il "canale di segnalazione" (SC). Il burst inizia con la sequenza CCR che consente alla stazione ricevente di stabilire con precisione la frequenza e la fase della portante, nonché il riferimento di clock per le decisioni sui dati. La parola unica che segue la sequenza CCR nel burst di riferimento viene utilizzata dalla generica stazione ricevente per ricostruire la temporizzazione di frame in modo da risalire alla posizione dei burst di traffico di proprio interesse entro la struttura del frame. Analogamente, la parola unica nel burst di traffico è utilizzata per segnalare la ricezione del burst stesso e per stabilire un riferimento temporale per la stazione ricevente che deve estrarre uno o più sottoburst di dati. La parola unica è normalmente composta da una sequenza di simboli binari caratterizzata da particolari proprietà di correlazione che ne rendono agevole e accurata la rivelazione. Nel ricevitore è presente un rivelatore della parola unica, che assume la forma di un correlatore digitale.

Tecnica CDMA

Questa importante tecnica, è nota come accesso multiplo a divisione di codice (CDMA). Essa deriva dalle comunicazioni a "spettro espanso" utilizzate in ambito militare a scopi "anti-jamming" e "anti-intercettazione". Nel CDMA, in cui ogni utente trasmette sempre e su tutta la banda. Non si hanno conflitti poiché ogni utente modula il proprio segnale con una sequenza pseudocasule, con determinate proprietà, ad una velocità molto maggiore della velocità di emissione dell'informazione (e quindi lo spettro del segnale viene sparpagliato su tutta la banda) mentre in ricezione, un messaggio viene isolato da un altro grazie alla proprietà di ortogonalità di ogni sequenza pseudocasuale rispetto alle altre, tramite una procedura di correlazione. In teoria, la coordinazione dinamica richiesta è pochissima, in contrasto con FDMA e TDMA in cui l'amministrazione di tempo e di frequenza ha un effetto diretto sulle prestazioni. Nei confronti delle tecniche di accesso più tradizionali, il CDMA presenta alcuni vantaggi quali una maggiore semplicità di allocazione di canali su domanda ed una migliore resistenza alle interferenze, mentre, d'altro canto, richiede una maggiore complessità sia dei circuiti di ricezione che degli algoritmi di decodifica e di sincronizzazione.

Tecnica SDMA

In questa tecnica i segnali sono discriminati nello spazio ed è la traiettoria di comunicazione che definisce i canali fisici. Un'architettura SDMA è basata fortemente su tecnologie d'antenna accoppiate con l'elaborazione avanzata dei segnali numerici. In contrasto con le applicazioni convenzionali in cui le posizioni sono illuminate costantemente da antenne a fascio fisso, nel sistema SDMA le antenne devono avere la capacità di illuminare le posizioni in modo dinamico. I fasci dell'antenna si devono dirigere elettronicamente e con elasticità verso l'utente in modo che, in una situazione ideale, la sua posizione da sola sia sufficiente per discriminarlo. ❑

CODICE MIP 2756793

luditronica

Le vignette ludiche di Fare Elettronica

ROBOTZone

di FEDERICO PINTO

Sempre più simili a noi! Costantemente i robot vengono modellati a somiglianza degli esseri umani, scopriamo il perché e dotiamo di una "testa" l'androide che stiamo creando!

TESTA robotica

Gli uomini comunicano con il linguaggio verbale, con quello corporeo, espressivo e gestuale mentre gli animali anche ad ultrasuoni, con sibili, richiami ed urla. E i robot? Oggi grazie a software appropriati possono collaborare tra loro, lavorare assieme ed essere "indipendenti", sempre sotto la stretta sorveglianza di programmi, sistemi di controllo e dell'uomo stesso. Al momento possono comunicare con l'essere umano tramite PC, tastiere, sistemi ad infrarossi, sensori ad ultrasuoni, GPS, joystick e chissà se un giorno arriveremo mai ad una vera e propria "intelligenza artificiale" dove i robot avranno propri pensieri e proprie emozioni e faranno delle scelte indipendentemente dall'essere umano. Fantascienza? Forse, ma la scienza, la tecnologia fanno passi da gigante e ogni anno gli investimenti in questo campo sono sempre maggiori; l'innovazione di materiali sempre più "intelligenti", la miniaturizzazione di motori ed ingranaggi e l'abbattimento dei costi ci fanno avere notizie di robot sempre più "umani" nella fluidità dei movimenti, nella forma e nelle dimensioni. Non passerà molto tempo che in ogni casa ci sarà un androide bipede a prendersi cura di noi, che conviveremo con macchine simili a noi, umanoidi che lavoreranno a fianco dell'uomo e che comunicheranno nel modo più conosciuto e comune, espressioni mimiche facciali, linguaggio verbale, vista, udito, riconoscimento visivo, vocale, ecc. L'essere umano ha creato una società intorno a sé di oggetti, strumenti e mezzi; una società dove le dimensioni e caratteristiche fisiche umane evolutive convivono alla perfezione; così, i robot che vengono creati si trovano di fronte un quotidiano costruito a dimensione di uomo e, poiché sarebbe impossibile cambiare la società per facilitare i robot, è più semplice investire in ricerca e creare robot simili a noi dotandoli di oggetti di nostro uso quotidiano. L'uomo crea robot per affermare in qualche modo la sua volontà di onnipotenza e soddisfare la sua brama di per-

Servocomandi utilizzabili figura 1 movimenti possibili del collo.

Esperimento di mimica facciale sfruttando la testa di una bambola.

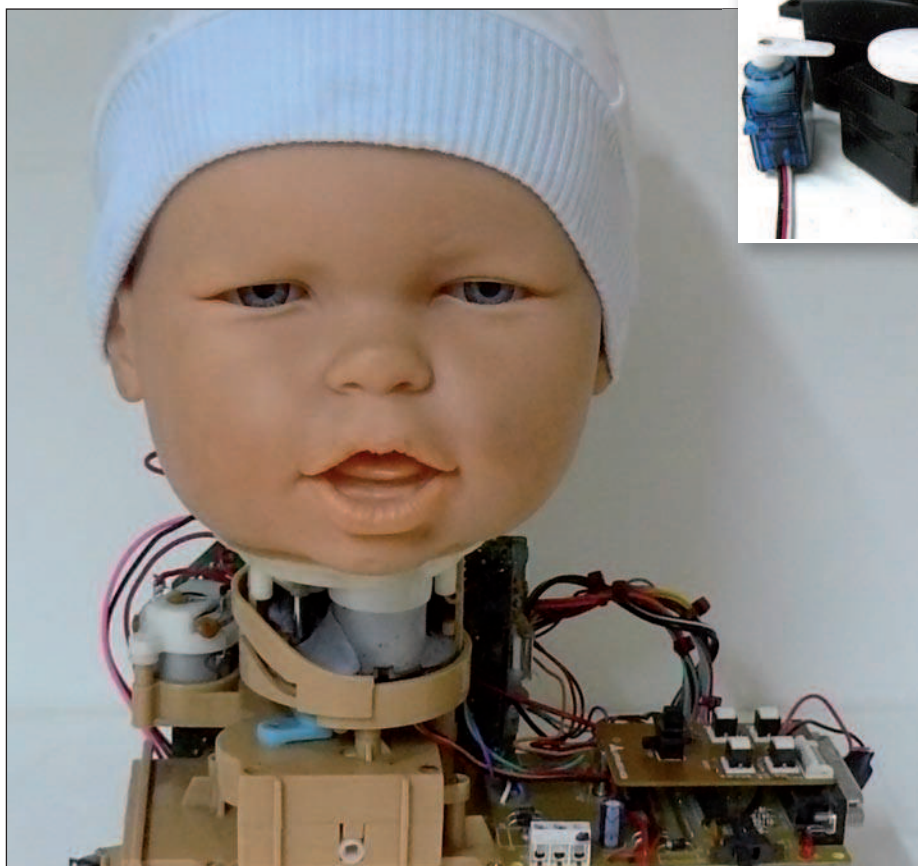
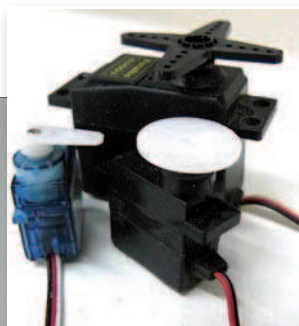
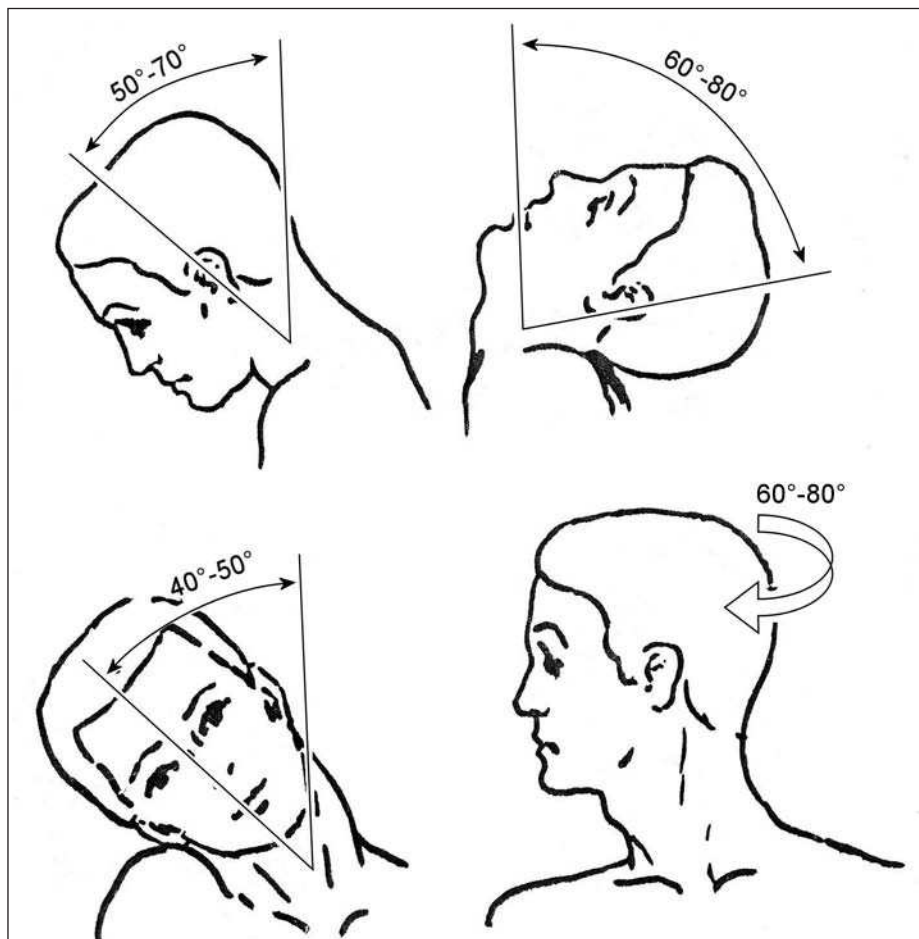



Figura 1: possibili movimenti del collo.

Figura 2: meccanica utilizzata per i movimenti del collo e del polso.

Figura 3: esempio virtuale di posizionamento attuatori.

fezione; in certi casi ci troviamo di fronte ad un vero e proprio scambio di personalità, causa la perfezione raggiunta. I centri di ricerca mondiali stanno investendo molto in questo, dotare cioè un androide di un sistema comunicativo e recettivo simile all'uomo, una "testa robotica" che possa comunicare, con tempi di risposta brevissimi (perchè così è abituato l'essere umano), i propri stati d'animo e d'umore, rispondere e reagire a stimoli esterni e chissà, magari in futuro, imparare e ricordare gli eventi a lui proposti ed imparare da eventuali errori. Tornando alla realtà e a quello che noi potremmo costruire nel nostro laboratorio, non avendo migliaia di euro da investire nella ricerca robotica per sofisticati sensori e sperimentali software, per dotare il nostro robot di una "testa" dovremmo adattarci da buoni amanti della meccanica e dell'elettronica al riciclo dei componenti e dei materiali. Prima di tutto preventivare e progettare la stessa (senza addentrarci in progetti faraonici); i risultati non saranno come quelli raggiunti dai centri di ricerca ma potremmo sempre costruire un volto che possa darci il buongiorno al mattino. Qualsiasi tipo di "testa robotica" abbiamo intenzione di costruire dobbiamo tener conto che sarà la prima cosa che verrà notata e che la persona che osserverà il vostro umanoide si aspetterà qualcosa di particolare. Psicologicamente l'uomo che tende a vedersi riprodotto, si aspetta un tipo di comunicazione simile alla sua, quindi espressiva e verbale. I movimenti principali per poter dar "vita" alla nostra testa meccanica li possiamo trovare nel collo e nel volto. I movimenti che possiede il collo e che tramite i muscoli trasmettono il movimento alla testa sono raffigurati in **figura 1**; essi daranno il movimento a tutta la testa mentre i muscoli facciali serviranno alla comunicazione e all'espressività. Come è possibile vedere in **figura 1**, i movimenti del collo sono riconducibili a quelli di abduzione/adduzione, flessione/estensione e rotazione; movimenti che abbiamo già riscontrato nella costruzione del "polso robotico". Potremmo riprodurre la stessa meccanica di movimento per far muovere la testa, tenendo conto che a fine costruzione la testa potrebbe avere un peso maggiore dei 300 grammi usati nel polso. Conviene quindi utilizzare attuatori che diano una certa affidabilità e sicurezza visto che dovranno sostenere tutto il peso della testa in movimento. Io ho utilizzato, per convenienza, dei micro cilindri ad aria con meccanica a giunto cardanico **(figura 2)**, ma la solita meccanica sarà possibile utilizzarla anche con dei motoriduttori elettrici e cavi che ne trasmettano il moto (possibilità di recuperare tutta la struttura, compresi i cavi e il motore, dal meccanismo dei vetri elettrici delle autovetture, ac-

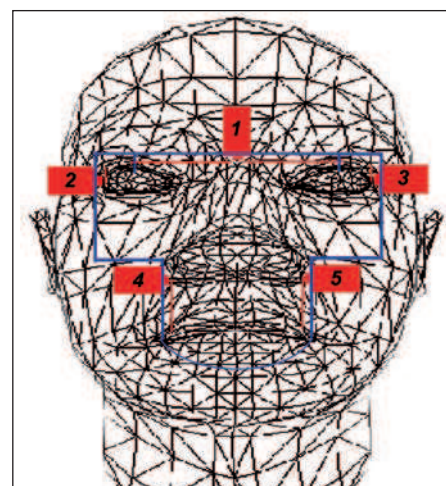
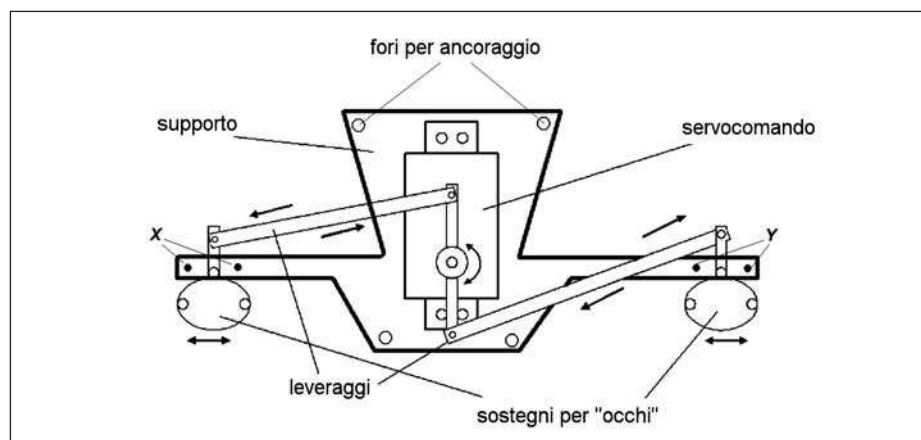



Figura 4: schema meccanico per movimento oculare dx/rx.

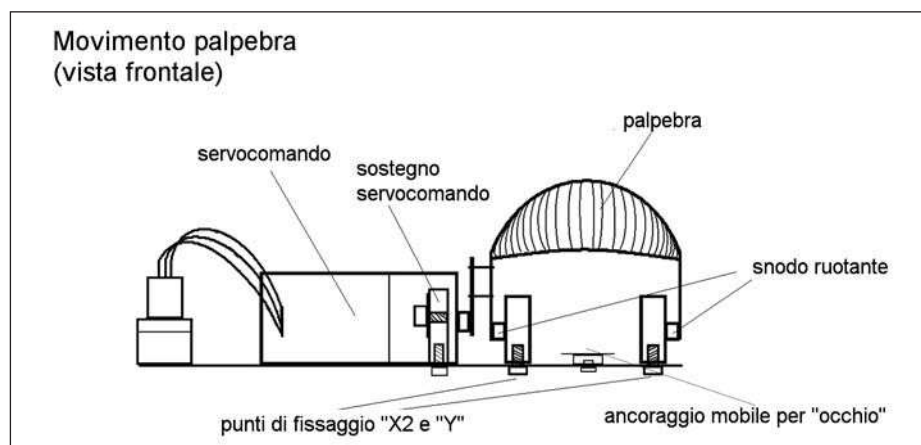



Figura 5: schema meccanico per chiusura apertura palpebre da abbinare a quello per il movimento oculare.

Figura 6: possibili "occhi" da usufruire.

Figura 7: protesi dentaria utilizzata per l'androide ancora in costruzione.

quistato in disfattura a 15 euro). Diciamo che la parte problematica potrebbe essere la costruzione dei movimenti espressivi facciali. Qui la cosa si fa complessa poiché il volto umano è composto da innumerevoli muscoli scheletrici (mandibolari), mimici (pellicciai), oculari, ecc... I mandibolari o masticatori collegano la scatola cranica alla mandibola, determinando i movimenti di quest'ultima: abbassamento, elevazione, proiezione anteriore, posteriore e laterale mentre gli altri muscoli sono innumerevoli e rappresentati da esili fasci muscolari che permettono l'espressività del volto. La loro riproduzione tutt'oggi non è semplice, basti pensare che per il movimento delle labbra vengo usati 12 muscoli e se alla nostra testa robotica installassimo 12 attuatori per il solo movimento delle labbra, le dimensioni, il peso e il controllo degli attuatori diventerebbero notevoli. Gli individui sono in grado di assumere a piacere un gran numero di espressioni facciali. Diverse ricerche sull'argomento concordano che le espressioni emozionali possono essere suddivise in sei grandi gruppi: paura, collera, felicità, sorpresa, tristezza, disgusto/disprezzo. Ma la perfezione umana può farci assumere ulteriori espressioni (oltre a quelle sopra citate) che il volto riesce a fare: emozioni non definite, come ad esempio interesse, dolore, perplessità, vergogna, stupore, ecc. Il volto umano può essere considerato senza alcun dubbio uno dei più potenti mezzi espressivi di comunicazione esistenti; infatti tutti noi siamo capaci di interpretare il pensiero e lo stato d'animo della persona con cui stiamo interagendo solamente osservando con attenzione l'espressione del viso. Proprio per la complessità della cosa, per ottenere un soddisfacente risultato nella costruzione meccanica dei movimenti facciali dovremmo ridurre ai minimi termini i movimenti (almeno per i primi esperimenti). Prima di tutto quali attuatori usare? I più idonei a questi tipi di movi-

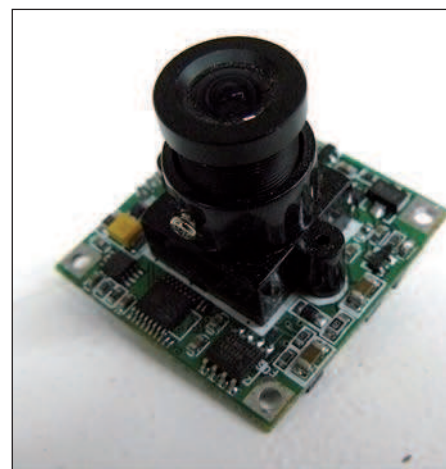
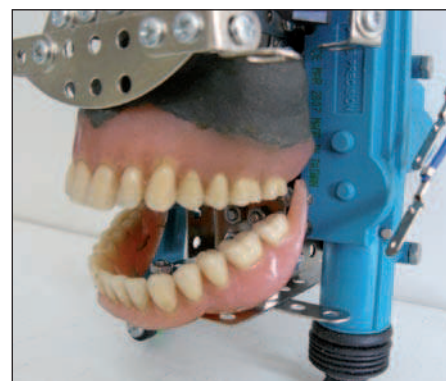
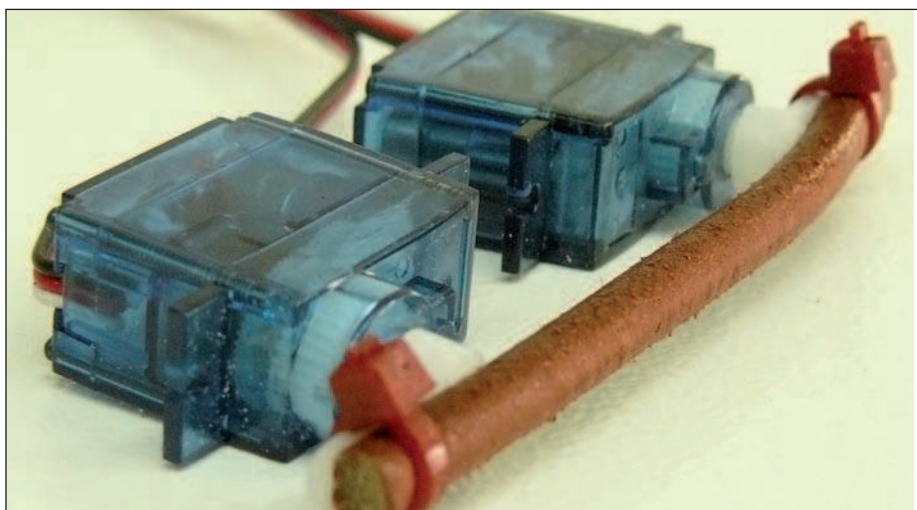



Figura 8: esempio di posizionamento del silicone ai due servocomandi.

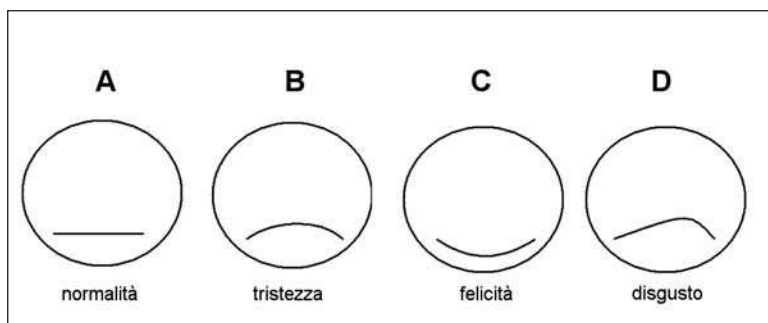
Figura 9: esempio di espressività ottenibile con due servocomandi.

menti facciali sono i servocomandi: ottima riduzione del moto, facile gestione, peso ridotto, di facile reperibilità e con varie tipologie di dimensioni e potenza (da non sottovalutare poi il fatto che siano già provvisti di leveraggi). Per quanto riguarda i movimenti facciali direi di iniziare da dei movimenti base: movimenti oculari, cioè movimento destra/sinistra dell'occhio e chiusura/apertura delle palpebre e movimento mandibolare, chiusura/apertura, da associare magari a dei messaggi vocali **(figura 3)**. Riportiamo in **figura 4** possibili progetti di movimento oculare utilizzando dei comuni servi. Chi è alle prime armi può ad esempio cimentarsi nella costruzione di questi semplici movimenti utilizzando comuni materiali, come bachelite o PVC. A vostro piacimento potrete applicare al movimento degli occhi videocamere, sensori di luce e di movimento, o rendere un po' più umana la vostra testa applicando un iride recuperato da un vecchio giocattolo **(figura 6)**. Nell'uomo, come spiegavo precedentemente, gran parte della comunicazione, oltre che di tipo espressivo e gestuale, è di tipo verbale. Per dar voce alla testa robotica, oggi in commercio esistono numerosi circuiti elettronici EPROM o supporti digitali, nei quali è possibile pre-registrare e memorizzare un gran numero di messaggi vocali (di una certa durata). Se il messaggio vocale pre-impostato viene attivato da sensori con una sorta di "intelligenza artificiale" e viene esternato da una bocca elettromeccanica, il suo effetto visivo sarà sicuramente affascinante e la nostra "testa robotica" prenderà sempre più una forma "umana". Nel mio specifico caso sono stato fortunato; avendo un amico odontoiatra, ho riciclato una protesi dentaria, che nel suo studio usava come protesi di prova. Niente di più perfetto per dar voce al mio umanoide. Se state osservando le protesi dentarie dei vostri nonni, cambiate idea, a loro servono! **(figura 7)** Potreste sempre chiedere presso uno studio odontoiatrico protesi o calchi dentari di prova, magari con pochi euro riuscirete a ottenere e costruire una "bocca" robotica perfetta. La chiusura della "bocca" da me utilizzata è comandata da due mini servocomandi e da del comunissimo filo di nylon che, richiamato dai leveraggi dei servi, fa serrare la bocca. Gli impulsi ai servocomandi (per il serraggio della parte inferiore della bocca)

Figura 10: animatronic bambino con movimenti mimici facciali.

possono essere inviati in sincronia con un messaggio vocale; in questo modo sembrerà che sia proprio la bocca ad esternare il messaggio. Se invece preferiamo far esprimere soltanto alcune emozioni al nostro robot come felicità o disgusto, potremmo servirci di due servocomandi, una striscia di gommapiuma o silicone ed alcune fascette da elettricisti. Il metodo è semplice; servendoci dei leveraggi del servo, fisseremo su di essi una striscia di gommapiuma o silicone e inviando impulsi elettrici ai servi potremmo comandarli contemporaneamente o singolarmente, ottenendo alcuni esempi come riportati in **figure 8** e **9**. Ancora una possibilità potrebbe essere quella di usare già una base su cui lavorare, riciclare cioè una testa di un giocattolo (nel mio caso ho usato la testa di un bambolotto) ed applicare internamente degli attuatori per ottenere i movimenti desiderati. Consiglio questo metodo a chi ha già un po' di esperienza in campo animatronico, visto che i siliconi usati per gli stampi dei giocattoli hanno un bassissimo spessore e lavorando su di essi si rischia di lacerare il silicone; per non parlare delle dimensioni interne da sfruttare che sono veramente ridotte ai minimi termini e poco spazio rimane da sfruttare.

Se però lo spazio fosse maggiore potremmo inserire internamente l'elettronica di controllo e gestione motori come se fosse un vero "cervello" **(figura 10)**. ❑

CODICE MIP 2756771

di FRANCO TEDESCHI e NICOLA DE CRESCENZO

PROGRAMMARE i LEGO MINDSTORMS *(parte sesta)*

TIMERS E *multitasking*

In questo numero parleremo dei timer, della gestione multi-tasking e della comunicazione tra robot

Fra le tante funzionalità, NQC mette a disposizione anche quattro timer, ovvero dei temporizzatori, che sono numerati da 0 a 3. Questi timer lavorano con unità temporali espresse in decimi di secondi e sono in grado di gestire valori da 0 a 32767, che corrispondono in termini di tempo, ad intervalli compresi tra zero e 55 minuti circa.
Vediamo come gestire questi strumenti: è possibile azzerare un timer con l'istruzione *ClearTimer(n)* ed ottenere il valore corrente con l'istruzione *Timer(n)*; il parametro n indica il numero del timer che intendiamo utilizzare.
Ecco subito un esempio pratico: il programma riportato nel **listato 1** fa accendere la lampada in dotazione al Lego Mindstorms per 5 secondi attraverso il *Timer (0).* I timer risultano molto utili al posto delle istruzioni *Wait()* per generare dei ritardi: a questo scopo, basta azzerare un temporizzatore ed aspettare che questo raggiunga un determinato valore.
La cosa interessante è che nel frattempo, il robot può anche reagire ad altri eventi, provocati ad esempio dai sensori. Il programma di esempio riportato nel **listato 2**, fa avanzare il robot in avanti finché non sono passati 15 secondi, oppure il sensore non tocca qualcosa. Questo comportamento viene ottenuto attraverso l'istruzione:

```
until((SENSOR_1 == 1)||(Timer(3) >150));
```

che attende finché la condizione tra parentesi non diventa vera.

LA GESTIONE MULTI-TASKING

Nel corso delle precedenti puntate, abbiamo visto che i programmi in NQC consistono di task (compiti); un task corrisponde ad un certo numero di istruzioni, chiamate anche statement. Un programma NQC può avere al massimo 10 task. Ogni task ha un suo nome, ma uno solo può essere denominato *main( )* ed è quello che viene eseguito automaticamente in fase di avvio. Gli altri task vengono invece eseguiti solamente se sono attivati con un apposito comando, denominato start. Una volta avviati, i task saranno eseguiti contemporaneamente. Un task in esecuzione può arrestare altri task usando il comando stop; successivamente un certo task può essere riavviato, ma dall'inizio e non dal punto in cui era stato fermato. A titolo di esempio, nel listato 3 riportiamo il codice di un programma che controlla il robot facendolo avanzare in avanti, in modo tale che quando urta un qualcosa, deve reagire cercando di aggirare l'ostacolo. Poiché il robot deve eseguire due compiti nello stesso momento, ossia avanzare e controllare i sensori, useremo un programma con due task. Il task principale si limita semplicemente ad impostare il tipo di sensore e ad attivare gli altri due task *leggi_sensor* e *move*. Il task *move* fa avanzare il robot in avanti; il secondo controlla se il sensore viene premuto e nel caso, compie le seguenti operazioni: arresta il task *move*, prende quindi il controllo del movimento del robot facendolo indietreggiare e voltare. Quindi fa nuovamente partire il task *move* per continuare ad avanzare. È importante ricordare che i task che facciamo partire vengono eseguiti in parallelo. Ciò è estremamente utile, ma può portare a risultati imprevisti, perché un task può anche interferire con gli altri, attraverso i cosiddetti effetti collaterali. Una maniera per risolvere questo problema è accertarsi che in ogni momento, un solo task stia controllando il robot; nel nostro caso infatti, *leggi_sensor* muove il robot solo dopo aver fermato *move*. Questo task non può quindi più interferire durante l'allontanamento dall'ostacolo; dopo aver indietreggiato, fa ripartire nuovamente *move*. Nonostante questa rappresenti una buona soluzione al problema, si pre-

Figura 1: porta IR del Mattoncino RCX

LISTATO 1

```
task main()
{
  ClearTimer(0);
  do
   {
     On(OUT_B);
    }
  while (Timer(0)<50);
  Off(OUT_B);
}
```

LISTATO 2

```
task main()
{
  SetSensor(SENSOR_1,SENSOR_TOUCH);
  ClearTimer(3);
  OnFwd(OUT_A+OUT_C);
  until ((SENSOR_1 == 1) || (Timer(3) >150));
  Off(OUT_A+OUT_C);
}
```

LISTATO 3

```
task main()
{
   SetSensor(SENSOR_1,SENSOR_TOUCH);
   start leggi_sensor;
   start move;
}
task move()
{
   while (true)
     {
       OnFwd(OUT_A+OUT_C);
       Wait(100);
      }
}

task leggi_sensor()
{
  while (true)
   {
     if (SENSOR_1 == 1)
      {
       stop move;
       OnRev(OUT_A+OUT_C);
       Wait(50);
       OnFwd(OUT_C);
       Wait(100);
       start move;
      }
   }
}
```

senta un'altra questione. Quando riavviamo *move*, questo riparte dall'inizio. Questa situazione è accettabile per piccoli task, ma spesso non rappresenta il comportamento più adeguato, nel senso che potremmo preferire fermare un task e quindi riprendere da dove lo avevamo sospeso.

I SEMAFORI

Una tecnica standard per risolvere il problema appena esposto, prevede l'uso di una variabile che segnala quando un task ha il controllo dei motori. Agli altri task non è quindi permesso accedere ai motori finché, il task che li aveva in uso, usando la variabile, non ne segnala la disponibilità. Una variabile di questo tipo viene chiamata semaforo. Vediamo subito un esempio applicativo! Indichiamo con *sem* la nostra variabile e supponiamo che il valore 0 indichi che nessun task sta correntemente facendo uso dei motori; quando un task vuole usare i motori, utilizzerà un protocollo di questo genere:

```
until (sem == 0);
sem = 1;
// Fai delle operazioni con i motori
sem = 0;
```

Con la prima istruzione, aspettiamo che nessuno stia usando i motori; ne dichiariamo quindi l'utilizzo impostando *sem* a 1 e a questo punto possiamo controllare i motori. Quando abbiamo finito, reimpostiamo *sem* a 0. Nel **listato 4** è riportata una versione del programma precedente, implementato con un semaforo. Quando il sensore urta qualcosa, il semaforo viene attivato e parte la procedura di allontanamento. Durante questa procedura, il task *move* è obbligato ad aspettare. Quando infine abbiamo terminato, il semaforo viene nuovamente posto a 0 cosicché *move* possa continuare. I semafori sono utili soprattutto per scrivere programmi complicati con task che operano in parallelo: un approccio al controllo di questo genere, garantisce che in ogni momento sia eseguita solo la parte critica di ogni task.

COMUNICAZIONE TRA I ROBOT

I Lego Mindstorms possono comunicare tra di loro usando la porta ad infrarossi; con questo sistema è possibile far interagire diversi robot assieme. Gli esempi applicativi possono essere tantissimi: dalle competizioni robotiche, alla costruzione di robot giganti che usano due o più RCX... il solo limite è la vostra fantasia! La comunicazione tra i robot avviene attraverso l'istruzione *SendMessage()*, che consente di inviare un valore numerico compreso tra 0 e 255 con gli infrarossi. Tutti gli altri robot posti nelle vicinanze, ricevono il messaggio e lo memorizzano. Un robot può ottenere il valore dell'ultimo messaggio ricevuto usando lo statement *Message()* e reagire di conseguenza. Affinché i robot possano comunicare correttamente tra loro è importante definire un cosiddetto protocollo di comunicazione, ovvero un insieme di regole che definiscono le modalità per lo scambio dei messaggi e la loro interpretazione. Premettiamo innanzitutto che quando si usano due o più robot, spesso uno è il leader e prende il nome di master. Gli altri robot sono invece slave. Il robot master invia i comandi agli slave e questi li eseguono; a loro volta gli slave possono rispondere al master, per indicare ad esempio il valore di un sensore.
Quando ci sono più slave, possiamo definire un protocollo in cui vengono inviati due codici (con una breve attesa frapposta): il primo indica il numero identificativo di uno slave, ed il secondo il comando vero e proprio.
Lo slave quindi eseguirà l'operazione solo se questa è destinata lui. Ciò richiede che ad ogni slave corrisponda un numero, operazione resa possibile dall'uso, ad esempio, di una costante che identifichi ogni robot. Nei **listati 5** e **6** è riportato un semplice esempio di comunicazione tra un master ed uno slave. Lo slave può obbedire a due di-

LISTATO 4

```
int sem;
task main()
{
  sem = 0;
  start move;
  SetSensor(SENSOR_1,SENSOR_TOUCH);
while (true)
{
  if (SENSOR_1 == 1)
  {
    until (sem == 0);
    sem = 1;
    OnRev(OUT_A+OUT_C);
    Wait(50);
    OnFwd(OUT_A);
    Wait(85);
    sem = 0;
   }
 }
}

task move()
 {
  while (true)
  {
    until (sem == 0);
    sem = 1;
    OnFwd(OUT_A+OUT_C);
    Wait(10);
    sem = 0;
           }
}
```

LISTATO 5

```
task main() // Slave
{
  while (true)
   {
     ClearMessage();
     until (Message() != 0);
     if (Message() == 1) {OnFwd(OUT_A+OUT_C);}
     if (Message() == 2) {Off(OUT_A+OUT_C);}
   }
           }
```

LISTATO 6

```
task main() // Master
{
SendMessage(1); Wait(200);
SendMessage(2); Wait(200);
}
```

versi ordini: avanzare e fermarsi. Il suo programma consiste di un semplice ciclo iterativo. In questo ciclo, imposta il valore del messaggio corrente a 0 usando l'istruzione *ClearMessage()*. Quindi aspetta finché questo non sia diverso da zero. A seconda del valore del messaggio, esegue uno dei due comandi. Il master si limita a spedire i messaggi per poi attendere un poco. Nel programma, ordina agli slave di avanzare, e poi, dopo due secondi, di fermarsi. dopo aver scritto questi due programmi, bisogna scaricarli sui robot. Attenzione! Ogni programma deve andare ad un solo robot. Quindi dobbiamo assicurarci di tenerne acceso uno soltanto durante l'operazione di download. Dopo aver completato questa operazione, accendiamo quindi entrambi i robot ed eseguiamo i programmi dello slave del master. Quando si gestiscono diversi robot, possono presentarsi due problemi:

1) se due robot inviano delle informazioni contemporaneamente, queste potrebbero andare perse. Per porre un rimedio a questa situazione, è importante definire nel protocollo di comunicazione, dei controlli, in maniera tale che, se una comunicazione fallisce, viene corretta. Per esempio, quando il master invia un ordine, dovrebbe ricevere una risposta di conferma dallo slave. Se non riceve una risposta entro un tempo ragionevole, ripete l'invio.

2) Inviare dei programmi dal computer a più robot senza interazioni indesiderate . Accertatevi sempre che, durante il download di un programma, sia acceso un solo robot! A questo proposito, è bene tener presente, che utilizzando più robot, è possibile fare in modo che solo il più vicino di questi, riceva il segnale. Ciò può essere ottenuto tramite l'istruzione *SetTxPower(TX_POWER_LO)* nel programma del master. Così facendo, il segnale IR viene inviato molto debolmente e solo un robot vicino al master può sentirlo. Ciò è particolarmente utile per costruire un robot più grande con due RCX. L'istruzione *SetTxPower(TX_POWER_HI)* permette di impostare nuovamente un lungo raggio di trasmissione. ❑

CODICE MIP 2765377

Lo shop della Community dell'elettronica

PRIMO PIANO

Spyder starter kit

Spyder Discovery Kit è un In-Circuit Debugger (ICD) specifico per microcontrollori Freescale MC9RS08KA, MC9S08QD e MC9S08QG, interfacciabile, via USB, con un normale PC basato su sistema operativo Windows. Con Spyder potrete effettuare il download del firmware all'interno della memoria del controllore grazie al tool software CodeWarrior Development Studio Special Edition (incluso) e sviluppare passo dopo passo le proprie applicazioni firmware, testandole in corso d'opera e monitorandone approfonditamente il funzionamento grazie al debugger integrato.

CODICE: USBSPYDER

PREZZO: € 9.90

Una raccolta di 200 progetti

NOVITA'

Una collezione di 200 idee di progetto pronte da utilizzare. Alimentatori e caricabatterie, sistemi di controllo, timers, radiantistica e telecomandi, sensori e strumentazione, audio e video. Sono queste le categorie dei progetti che troverete in questo nuovo fantastico fascicolo! Il nuovo numero di Fare Elettronica EXTRA "200 progetti per 2000 applicazioni" sarà in edicola a soli 6 Euro nel mese di Settembre, ma gli abbonati a Fare Elettronica potranno già ordinarlo al prezzo speciale di 4 Euro!

CODICE: FE-ABB SPECIALE 200 PROGETTI

PREZZO: € 6.00

NOVITA'

ProfiLab-Expert

Il software ProfiLab-Expert permette di sviluppare i propri progetti di misurazione digitale o analogica. Non importa se si vuole creare misurazioni analogiche o controlli digitali - è possibile realizzare qualsiasi cosa senza dover scrivere una sola riga di codice programma. Si creano i propri progetti così come si disegnano diagrammi di connessione. In modo semplice, si aggiungono pulsanti, displays etc al proprio diagramma costruendo il progetto passo dopo passo. ProfiLab-Expert è una potente combinazione di DMM-ProfiLab e Digital-ProfiLab. Il software ProfiLab-Expert offre tutte le caratteristiche dei singoli prodotti e molte altre funzioni quali il compilatore integrato il supporto hardware esteso.

CODICE: PROFILAB-EXPERT 4.0

PREZZO: € 119.88

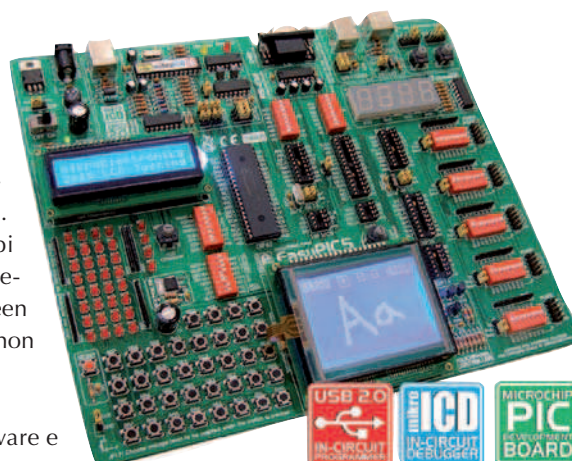
easyPIC5

La scheda di sviluppo EasyPIC5 supporta i microcontrollori PICmicro a 8,14, 18, 20, 28, and 40-pin. Ogni jumper, elemento o pin è chiaramente marcato sulla scheda. Grazie a EasyPIC5 è possibile sviluppare e testare la maggior parte delle applicazioni industriali: temperatura, controlli, contatori, timers, ecc.

OFFERTA

Comprese nel prodotto vengono fornite applicazioni ed esempi pratici in PASCAL, BASIC, C. La scheda consente di eseguire il debug IN-CIRCUIT in tempo reale. E' possibile collegare il touch screen (opzionale) al display grafico 128x64 quindi utilizzare la periferica non solo per visualizzazioni grafiche ma come dispositivo di input.

La confezione contiene:

1 scheda easyPIC5 - 1 cavo seriale - 1 cavo USB - 1 CD con software e manuale - Manuali cartacei.

CODICE: EASYPIC5

PREZZO: ~~€ 154.80~~ € 118.80

MikroC-PRO

Compilatore C per PIC. La funzionalità mikroICD costituisce uno strumento altamente efficiente per il debugging in Real-Time a livello hardware. Il debugger mikroICD permette di eseguire il programma mikroC PRO for PIC 2009 su un microcontrollore host PIC e vedere i valori di variabili, Registri (SFR), RAM, CODE e memoria EEPROM insieme all'esecuzione del codice mikroICD sull'hardware. Se possedete un appropriato hardware e software per utilizzare mikroICD, allora dovete scegliere tra Release build type o ICD Debug build type appena finita la fase di programmazione del vostro programma.

OFFERTA

CODICE: MIKROC-PRO

PREZZO: ~~€ 190.80~~ € 142.80

BEST SELLER

CB380, un PLC compatto

Modulo core CUBLOC basic con controller ladder logic. Ideale per applicazioni che richiedono un modesto numero di porte I/O, per piccole e grandi produzioni (da 1 a 1000), integrabile in PCB personalizzati disponibili per produzione di massa (1,000+).

- Supporto MODBUS ASCII (Master & Slave Mode).
- Supporto MODBUS RTU (Master & Slave Mode).
- Periferiche esterne CuNET (I2C) incluse.
- Periferiche esterne SPI incluse.
- Sviluppo attraverso CublocStudio in BASIC e/o linguaggio Ladder Logic.

CODICE: CB380

PREZZO: € 70.80

Lo shop della Community dell'elettronica

3 arretrati ad un prezzo speciale

BUNDLE

Tre arretrati di Fare Elettronica a tua scelta ad un prezzo veramente vantaggioso! Se hai perso qualche numero approfitta subito di questa opportunità per completare la tua collezione di Fare Elettronica.

CODICE: BUNDLE B

PREZZO: € 15.00

Display alfanumerico con interfaccia seriale

Display alfanumerico 16x2 con retroilluminazione verde (64.5 x 13.8 mm). Sul retro del CLCD una scheda di controllo riceve i segnali I2C o RS232 e li visualizza sul display. Questa scheda può comunicare con i protocolli I2C e RS232.
L' RS232 supporta i seguenti baudrate: 2400, 4800, 9600, 14400, 19200, 38400, 57600, 115200, 8-N-1. Sono disponibili 2 connettori seriali, uno per i segnali a 5V (3pin) e l'altro per i segnali a +/- 12V (4pin).

CODICE: CLCD216-G

PREZZO: € 50.40

Database di componenti ECA

Non c'è bisogno di tante parole per presentare i data-book della ECA, ogni laboratorio che si rispetti ne possiede uno. Al passo con i

tempi, la famosa società Tedesca, ha riportato su CDROM i suoi prodotti più famosi; sono prodotti di alta qualità che richiedono risorse minime, quindi sono utilizzabili anche su vecchi PC (minimo PC AT486, 8 MB ram, Windows 3.1/95/98/NT/2000/XP/VISTA). Sviluppato un software completamente nuovo con un menu semplificato, la rappresentazione dei datasheet uniforme, un nuova struttura del data-base. Inoltre ora permette una selezione dei parametri dei transistori, i diodi, i tiristori ed i FETs. Mediante la "Funzione-Note" è possibile aggiungere i dati personali a ciascu dispositivo. Il reference manual, già popolare nella versione data-book, permette una veloce cernita delle informazioni più dettagliate.
Il Data-base contiene i dati dei componenti elettronici come circuiti integrati, i transistori, i diodi, i FETs, i tiristori, UJTs con tutti i parametri più importanti e i datasheet in formato PDF dal server della ECA.

Contenuti:
oltre 90.000 Diodi
oltre 55.000 FETs
oltre 130.000 Transistor
oltre 18.000 Tiristori
oltre 105.000 circuiti integrati
Divisi in differenti Datasheet.
Il nuovo DVD include il DataBase del TDV-Disk e del DDV-Disk.

CODICE: ECA-403-09

PREZZO: € 54,00

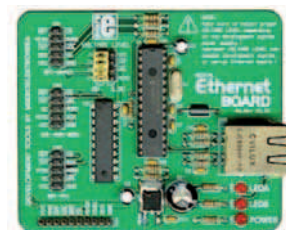
KIT per esperimenti

Una utilissima collezione di componenti e circuiti integrati: Generatore di Impulsi, Breadboard, Serial EEPROMs, sensori di temperatura, Real Time Clock, RAM seriali, Dallas Touch Button, chip per espansione I/O, potenziometro elettronico seriale, Oscillatori a 4MHz e 10MHz, fili per interconnessioni. Allegato al kit viene fornito uno schemario per utilizzare i vari componenti.

CODICE: EXP-KIT

PREZZO: € 88.80

Aggiungi Ethernet alla tua applicazione!

La rete locale LAN (Ethernet) è la tecnologia più comune utilizzata oggi. A bordo della scheda le stazioni comunicano l'un l'altro trasmettendosi i pacchetti di dati. La scheda EP-S-ETHERNET è dotata di un MAC address a 48 Bit, che è

utilizzato per specificare sia la destinazione che la fonte di ogni pacchetto dati. La Serial Ethernet Board ha un controller Ethernet a 28 pin (ENC28J60) 10 BASE-T con un Media Access Control e un Physical Layer (MAC & PHY), memoria RAM (buffer) a 8Kbytes e Serial Peripheral Interface (SPI).

CODICE: EP-S-ETHERNET

PREZZO: € 34.80

www.ieshop.it

Ordinare direttamente tramite internet conviene! Potrai infatti aggiudicarti dei buoni sconto (coupon) con le modalità:

COUPON ABBONAMENTO ONLINE FE E FW
Sottoscrivendo un nuovo abbonamento o rinnovando il proprio online, direttamente tramite IE SHOP, si ha diritto ad un coupon del 15%! **Qualora ci si abbona (sempre online) a Fare Elettronica e Firmware insieme il coupon è del 20%!**

COUPON QUIZ "ELETTROQUIZ"
Rispondendo ai Quiz presente ogni mese su Fare Elettronica si avrà diritto ad un coupon del 10%. Tutte le risposte pervenute saranno accuratamente valutate dagli autori dei quesiti e, in caso di risposta esatta (oltre al Coupon) potrai vincere fantastici premi!

COUPON "ACQUISTI PREMIATI"
I tuoi acquisti su www.ieshop.it vengono premiati con un coupon del 10% se il tuo ordine supera i 100 EUR (IVA e spese di trasporto escluse). Ma se il tuo ordine supera i 200 EUR, lo sconto sale al 20%!

IMPORTANTE!
- Il coupon può essere speso **esclusivamente** ordinando sul sito www.ieshop.it
- Il coupon non potrà essere utilizzato in caso di nuova sottoscrizione o rinnovo dell'abbonamento, è invece utilizzabile per tutti gli altri prodotti presenti su www.ieshop.it
- Ogni coupon ha una scadenza, non dimenticarlo!
- Lo slogan "più acquisti più risparmi" pecca senz'altro di originalità, ma in questo caso è molto azzeccato: i coupon sono infatti utilizzabili solo una volta, usalo bene!
- I coupon non sono cumulabili, e vanno utilizzati singolarmente, secondo il criterio cronologico di assegnazione.

CONTI CORRENTI POSTALI - Ricevuta di Accredito — BancoPosta

€ sul C/C n. 70107552 di Euro

TD 451 — importo in lettere

INTESTATO A: INWARE EDIZIONI SRL

CAUSALE

ESEGUITO DA: VIA - PIAZZA, CAP, LOCALITÀ

AUT. DB/SISB/E 20017 DEL 11.04.2006

BOLLO DELL'UFF. POSTALE codice bancoposta

IMPORTANTE: NON SCRIVERE NELLA ZONA SOTTOSTANTE — importo in euro, numero conto, td

70107552 < 451>

CONTI CORRENTI POSTALI - Ricevuta di Versamento — BancoPosta

€ sul C/C n. 70107552 di Euro — importo in lettere

INTESTATO A: INWARE EDIZIONI SRL

ESEGUITO DA: VIA - PIAZZA, CAP, LOCALITÀ

BOLLO DELL'UFF. POSTALE

AVVERTENZE
Il Bollettino deve essere compilato in ogni sua parte (con inchiostro nero o blu) e non deve recare abrasioni, correzioni o cancellature.
La causale è obbligatoria per i versamenti a favore delle Pubbliche Amministrazioni.
Le informazioni richieste vanno riportate in modo identico in ciascuna delle parti di cui si compone il bollettino.

CAUSALE

IMPORTANTE: NON SCRIVERE NELLA ZONA SOPRASTANTE!

AVVERTENZE

Il Bollettino deve essere compilato in ogni sua parte (con inchiostro nero o blu) e non deve recare abrasioni, correzioni o cancellature.

La causale è obbligatoria per i versamenti a favore delle Pubbliche Amministrazioni.

Le informazioni richieste vanno riportate in modo identico in ciascuna delle parti di cui si compone il bollettino.

Indagine sui lettori

Aiutaci a conoscerti meglio!
Con il tuo aiuto riusciremo ad offrirti una rivista sempre più in linea con le tue aspettative!

Nome ....

Cognome ....

Via .... n° ....

Cap .... Città .... Prov ....

Email ....

IL TUO SETTORE DI COMPETENZA:

- ❑ **B05** Direzione Tecnica
- ❑ **B06** Progettazione
- ❑ **B07** Studente
- ❑ **B08** Direzione Acquisti
- ❑ **B09** Insegnante
- ❑ **B10** Altro . . . .

PRODOTTO PRINCIPALE O SERVIZIO OFFERTO DALL'AZIENDA DOVE LAVORI:

- ❑ **C11** Apparecchiature elettriche, elettroniche, ICT
- ❑ **C12** Elettrodomestici
- ❑ **C13** Consulenza
- ❑ **C14** Apparecchiature scientifiche, misura e controllo
- ❑ **C15** Automotive
- ❑ **C16** Vending
- ❑ **C17** Altro . . . .

NUMERO DI DIPENDENTI DELLA TUA AZIENDA:

- ❑ **D18** fino a 10
- ❑ **D19** da 10 a 50
- ❑ **D20** da 50 a 100
- ❑ **D21** da 100 a 500
- ❑ **D22** oltre 500

FE

Solo se sei abbonato, indica il tuo codice abbonato: . . . .
e barra la casella di interesse:

TIPO DI ABBONAMENTO:

- ❑ **A01** Personale uso professionale
- ❑ **A02** Aziendale
- ❑ **A03** Scuola o Università
- ❑ **A04** Personale uso hobbistico

Compila il cedolino e invialo
in busta chiusa o via fax allo 02 66508225
e riceverai GRATIS a tua scelta
UN CD-ROM DEL VALORE DI 10 EURO

FOTOGRAFIA DIGITALE: un corso completo che guida l'utente nel mondo della fotografia utilizzando le moderne fotocamere digitali.

BABYLON: il traduttore istantaneo italiano/inglese inglese/italiano più conosciuto nel mondo.

iPOD CONVERTER: il software per convertire i filmati e i file audio in un formato idoneo all'iPOD.

Grazie per la preziosa collaborazione!

Se vuoi ordinare "offline" ovvero senza usare internet, usa il seguente modulo d'ordine **PRODOTTI**

COD. PRODOTTO	DESCRIZIONE	PREZZO UNITARIO	Q.tà	Totale
			SUB TOTALE	
			SPESE DI SPEDIZIONE	
			TOTALE	

Nota. Se le righe non sono sufficienti, utilizzare più copie del modulo.
Per maggiori dettagli sulle spese di spedizione e i metodi di pagamento vedi retro cartolina.

METODI DI PAGAMENTO

☐ **BONIFICO BANCARIO** ☐ **BOLLETTINO POSTALE** ☐ **CONTRASSEGNO**

☐ EUROCARD MasterCard ☐ VISA ☐ AMERICAN EXPRESS

TITOLARE. . . . .
NUMERO . . . .
DATA DI SCADENZA. . . . . CODICE DI SICUREZZA . . . .

INVIARE IL MODULO D'ORDINE A: Inware Edizioni srl, via Cadorna 27/31 - 20032 Cormano (MI) - oppure via FAX allo 02 66508225
(Inviando questo modulo si accettano i termini e le condizioni riportate a tergo)

Nome . . . . Cognome . . . .
Indirizzo . . . . n° . . . .
Cap . . . . Città . . . . Prov . . . .
Tel . . . . Fax . . . . Email . . . .
Ragione Sociale . . . .
Codice fiscale . . . . Partita IVA . . . .

DATA . . . . FIRMA . . . . ☐ Fattura

Sì, mi abbono alle riviste di Inware Edizioni

FE

ABBONAMENTO STANDARD
Una rivista € 49,50
Due riviste € 89,50
Tre riviste €119,50

ABBONAMENTO PLUS (include il CD dell'annata 2008)
Una rivista € 55,50 (UN CD)
Due riviste € 99,50 (DUE CD)
Tre riviste €134,50 (TRE CD)

SELEZIONE DELLE RIVISTE
Una rivista ☐ FARE ELETTRONICA ☐ FIRMWARE ☐ ELEKTOR
Due riviste ☐ FARE ELETTRONICA ☐ FIRMWARE ☐ ELEKTOR
Tre riviste ☐ FARE ELETTRONICA ☐ FIRMWARE ☐ ELEKTOR
Indicare con una croce la preferenza

METODI DI PAGAMENTO (per maggiori dettagli vedi retro cartolina)

☐ **BONIFICO BANCARIO** ☐ **BOLLETTINO POSTALE** ☐ **ALLEGO ASSEGNO** (Intestato a Inware Edizioni)

☐ EUROCARD MasterCard ☐ VISA ☐ AMERICAN EXPRESS

TITOLARE. . . . .
NUMERO . . . .
DATA DI SCADENZA. . . . . CODICE DI SICUREZZA . . . .

INVIARE IL MODULO D'ORDINE A: Inware Edizioni srl, via Cadorna 27/31 - 20032 Cormano (MI) - oppure via FAX allo 02 66508225
(Inviando questo modulo si accettano i termini e le condizioni riportate a tergo)

Nome . . . . Cognome . . . .
Indirizzo . . . . n° . . . .
Cap . . . . Città . . . . Prov . . . .
Tel . . . . Fax . . . . Email . . . .
Ragione Sociale . . . .
Codice fiscale . . . . Partita IVA . . . .

DATA . . . . FIRMA . . . . ☐ Fattura

ISTRUZIONI PER L'ORDINE E SPESE DI SPEDIZIONE

Il modulo d'ordine dovrà essere compilato in tutte le sue parti ed inviato via posta o via fax ai recapiti indicati sul modulo d'ordine stesso. Gli ordini potranno essere fatti anche direttamente online dal sito www.ieshop.it. In questo caso non sarà necessario inviare il modulo d'ordine. Le spese di spedizione ammontano a euro 8,50 a cui vanno aggiunti euro 3,50 se si sceglie di pagare in contrassegno.

METODI DI PAGAMENTO

Si accettano pagamenti ino in contrassegno, carta di credit, bollettino postale o bonifico all'ordine. Per il contrassegno verrà applicata una spesa aggiuntiva di euro 3,50 per le spese di contrassegno. Forme diverse di pagamento devono essere previamente concordate.

Bonifico bancario

Appoggiato su Poste Italiane IBAN: IT 68 I 07601 01600 000070107552 intestato a Inware Edizioni srl.

Bollettino postale

Versamento sul conto corrente postale n. 70107552 intestato a Inware Edizioni srl. Inviare la ricevuta (o copia) del versamento.

Contrassegno

La merce verrà pagata direttamente al corriere alla consegna della merce. Il pagamento in contrassegno comporta l'addebito di euro 3,50per spese di contrassegno.

Carta di Credito

Il pagamento con carta di credito può essere effettuato specificando i dati della carta via posta, email, web, fax o telefono. Per gli ordini effettuati sul sito il pagamento mediante carta di credito verrà effettuato attraverso una connessione sicura SSL per garantire la massima sicurezza.

TERMINI E CONDIZIONI

Tutti i prodotti sono coperti da garanzia. La garanzia copre tutti i difetti di fabbricazione entro un anno dal ricevimento della merce. Tutti i prodotti non funzionanti per uso improprio o incuria non saranno ritenuti in garanzia e saranno addebitati gli eventuali costi di riparazione. Tutti i prodotti verranno riparati e/o sostituiti direttamente dal produttore. Non sono coperti da garanzia i componenti elettronici (microprocessori, memorie, ecc.) La garanzia dei prodotti si intende F.co ns. sede, le eventuali spese di trasporto sono a carico del cliente salvo accordi diversi. Per dar corso alla riparazione/sostituzione in garanzia è necessario seguire l'apposita procedura di RMA.

PRIVACY

Ai sensi del Decr. Lgs. 196/2003 la informiamo che i dati trasmessi verranno impiegati coi principali scopi di indagini di mercato e nelle modalità previste dallo stesso, prevalentemente con mezzi informatici. Il conferimento, di norma facoltativo, è obbligatorio per permettere il rapporto commerciale. È in ogni caso fatto diritto dell'interessato esercitare i propri diritti, nei modi previsti dal "Titolo II art. 7" della legge sopra citata, scrivendo a Inware Edizioni srl, Via Cadorna 27 – 20032 Cormano o tramite email all'indirizzo info@inwaredizioni.it

---

ABBONARSI ALLE RIVISTE INWARE EDIZIONI CONVIENE!

I vantaggi per gli abbonati sono, oltre al prezzo bloccato per un anno, la ricezione del numero direttamente a casa con la garanzia di ricevere tutti i numeri.
Inoltre un vistoso risparmio che nel caso dell'abbonamento a tre riviste, ammonta a ben 58 euro.
L'abbonamento ha una durata di 12 mesi e comporta l'invio di 11 numeri di Fare Elettronica. Eventuali variazioni di indirizzo andranno comunicate tempestivamente alla Redazione che provvederà a registrare il cambiamento senza alcuna spesa aggiuntiva. L'abbonamento decorrerà dal primo numero raggiungibile alla data di avvenuto pagamento. Non sono previsti rimborsi in caso di disdetta dell'abbonamento.

METODI DI PAGAMENTO

Bonifico bancario

Appoggiato su Poste Italiane IBAN: IT 68 I 07601 01600 000070107552 intestato a Inware Edizioni srl.

Bollettino postale

Versamento sul conto corrente postale n. 70107552 intestato a Inware Edizioni srl. Inviare la ricevuta (o copia) del versamento indicando nella causale:
"Abbonamento Fare Elettronica"

Carta di Credito

Il pagamento con carta di credito può essere effettuato specificando i dati della carta via posta, email, web, fax o telefono. Per gli ordini effettuati sul sito il pagamento mediante carta di credito verrà effettuato attraverso una connessione sicura SSL per garantire la massima sicurezza.

Assegno bancario

E' possibile spedire un assegno bancario insieme a questo coupon.